BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF BUSINESS AND MANAGEMENT
FAKULTA PODNIKATELSKÁ

INSTITUTE OF INFORMATICS
ÚSTAV INFORMATIKY

APPLICATION OF SCRUM METHODOLOGY ON A SOFTWARE DEVELOPMENT PROJECT

NASAZENÍ METODIKY SCRUM PŘI VÝVOJI SOFTWARE

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE
Bc. Matúš Burzala

SUPERVISOR
VEDOUCÍ PRÁCE
Ing. Lenka Smolíková, Ph.D.

BRNO 2021

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF BUSINESS AND MANAGEMENT

Specification Master's Thesis

Department:	Institute of Informatics
Student:	**Bc. Matúš Burzala**
Study programme:	System Engineering and Informatics
Study field:	Information Management
Supervisor:	**Ing. Lenka Smolíková, Ph.D.**
Academic year:	2020/21

Pursuant to Act no. 111/1998 Coll. concerning universities as amended and pursuant to the BUT Study Rules, by the Director of the Institute, you have been assigned a Master's Thesis entitled:

Application of SCRUM Methodology on a Software Development Project

Characteristics of thesis dilemmas:

Introduction
Goals of thesis and methods
A theoretical review of a problem
Analysis of the contemporary situation
Proposal of solution
Conclusion
References
Appendixes

Objectives which should be achieve:

The aim of the thesis is an application of SCRUM methodology on a software development project in order to optimize the working process.

Basic sources of information:

BECK, et al. Manifesto for Agile Software Development [Online]. 2001 [cit. 2021-02-18]. Retrieved from: https://agilemanito.org/iso/en/manifesto.html

MYSLÍN, J. Scrum: průvodce agilním vývojem softwaru. Brno: Computer Press, 2016. ISBN 978--251-4650-7.

SCHWABER, K. and J. SUTHERLAND. The Scrum Guide [Online]. 2020 [cit. 2021-02-18]. Retrieved from: https://scrumguides.org/

Scrum.org - The Home Of Scrum [Online]. 2021 [cit. 2021-02-18]. Retrieved from: https://www.scrum.org/

ŠOCHOVÁ, Z. and E. KUNCE. Agilní metody řízení projektů. Brno: Computer Press, 2014. ISBN 978-8-251-4194-6.

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2020/21

In Brno dated 28.2.2021

L. S.

Mgr. Veronika Novotná, Ph.D. Director of the Institute	doc. Ing. Vojtěch Bartoš, Ph.D. Dean

Abstract

The diploma thesis deals with the comparison of the methodology used on the software development project with the SCRUM methodology. It maps all roles, events, and artifacts of the project and identifies their differences from the SCRUM definition. The thesis also contains the proposal of what needs to be adjusted or changed to reach the correct application of the SCRUM and therefore the optimization of the development process.

Abstrakt

Diplomová práca sa zaoberá porovnaním metodiky použitej na projekte vývoja software a metodiky SCRUM. V rámci práce sú zmapované všetky role, udalosti a artefakty projektu u ktorých sú následne identifikované ich odlišnosti od definície metodiky SCRUM. Práca ďalej obsahuje návrh toho, čo je potrebné upraviť, alebo zmeniť, aby sa dosiahla správna aplikácia metodiky SCRUM a tým pádom aj optimalizácia vývojového procesu.

Keywords

agile, agile methodologies, Scrum, software development

Klíčová slova

agilný vývoj, agilné metodiky, Scrum, vývoj softvéru

Reference

BURZALA, Matúš. *Application of SCRUM Methodology on a Software Development Project.* Brno, 2021. Master's thesis. Brno University of Technology, Faculty of Business and Management. Supervisor Ing. Lenka Smolíková, Ph.D.

Rozšířený abstrakt

Diplomová práca sa zaoberá aplikáciou agilnej metodiky SCRUM na projekt vývoja softvéru. Konkrétne sa jedná o projekt firmy so sídlom v Brne, ktorá je jednou zo štyroch pobočiek materskej firmy so sídlom v Škandinávii. Materská spoločnosť, rovnako ako aj všetky jej dcérske spoločnosti sa zaoberá vývojom rozličných softvérových riešení, ktoré nachádzajú uplatnenie v sektoroch ako financie, zdravotníctvo, automobilový priemysel, energetika, maloobchod, telekomunikácie a mnoho ďalších.

Projektom, ktorý je predmetom tejto práce, sa rozumie vývoj modulu na správu siete užívateľských dát, ktorý je súčasťou uceleného produktu zloženého z viacerých softvérových modulov ako aj hardvérových prostriedkov dodávaných ako jeden produkt v rámci materskej spoločnosti. Na vývoji sa v rámci predmetného projektu podieľajú štyri vývojové tými, lokalizované v Brne, z ktorých niektorí členovia pracujú permanentne na diaľku. Každý tím sa skladá zo šesť až deväť vývojárov, jedného člena v role Scrum Master a v rámci celého projektu ďalej existujú dvaja zamestnanci v role Product Owner a jeden zamestnanec na pozícii Systémového Architekta. Je nutné podotknúť, že v rámci práce sa pod pojmom vývojár rozumie technicky zameraný zamestnanec, podieľajúci sa na vývoji produktu, ktorého práca zahrňuje činnosti ako analýza požiadaviek, návrh riešenia, tvorba zdrojového kódu, tvorba testovacieho kódu, manuálne testovanie, konfigurácia sieťových topológií a mnohé iné. Ďalším faktom je to, že aktuálne používaná metodika v rámci projektu je založená na metodike SCRUM, čomu odpovedajú aj názvy rolí ako Scrum Master a Product Owner. Tak isto ako u metodiky SCRUM, vývoj je v rámci projektu rozdelený do po sebe sa opakujúcich cyklov nazývaných Sprint a taktiež sú v rámci cyklov uskutočňované udalosti podobné udalostiam metodiky SCRUM. Je však nutné podotknúť, že aktuálny stav odpovedá metodike SCRUM len v niektorých rysoch, čo potvrdzuje aj analytická časť práce a celkový obsah práce a v ňom uvedené skutočnosti. Cieľom práce je teda identifikácia nedostatkov a odlišností aktuálne používanej metodiky v rámci predstaveného projektu, ako aj návrh riešenia na ich odstránenie, ktorého výsledkom by malo byť zefektívnenie procesu vývoja.

Práca je rozdelená do troch častí, z ktorých prvá, teoretická časť, obsahuje popis životného cyklu vývoja softvéru a jeho konkrétnych fáz. V rámci tradičných modelov vývoja softvéru sú v práci predstavené štyri modeli, Vodopádový model, Iteratívny model, Špirálový model a V-model vývoja softvéru. Teoretická časť ďalej pokračuje úvodom do agilných metód vývoja softvéru, ktorý v krátkosti popisuje podstatu agilných metód a vyzdvihuje ich kľúčové vlastnosti. V rámci agilných modelov bol do práce zahrnutý detailnejší popis modelov Crystal, Extreme Programming, Lean Software Development a Kanban. U všetkých uvedených agilných, ale aj tradičných modeloch vývoja softvéru je uvedený ich stručný popis, grafické zobrazenie modelu a zoznam ich výhod a nevýhod vo vzťahu k použitiu modelu na projekty s konkrétnymi vlastnosťami. Posledným modelom predstaveným v rámci teoretickej časti práce je SCRUM. Popis tohto modelu je vypracovaný na detailnejšej úrovni v porovnaní s ostatnými modelmi. Najprv je predstavená podstata modelu a následne je v práci uvedený rozbor tímovej štruktúry a konkrétnych rolí, ktoré sú Produc Owner, Scrum Master a Developer. U každej role je uvedený jej význam a zodpovednosti v kontexte projektu. Po predstavení rolí nasleduje špecifikácia udalostí ako Sprint, Sprint

Planning, Daily Scrum, Sprint Review a Sprint Retrospective. Každá udalosť má špecifikovaný časový rámec konania, zoznam účastníkov a ich úloh v rámci udalosti, agendu a hlavne cieľ, za účelom naplnenia ktorého je daná udalosť organizovaná. Všetky uvedené udalosti okrem Sprint, majú tiež určené svoje umiestnenie v rámci Sprint. Okrem toho je u teoretického popisu metodiky SCRUM uvedený ešte zoznam artefaktov, ktoré sa v rámci neho používajú, a to konkrétne Product Backlog, Sprint Backlog a Incremet. Každý artefakt obsahuje pridružený záväzok, ktorým je konkrétne u Product Backlog - Product Goal, Sprint Backlog - Sprint Goal a u Increment – Definition of Done. Popis každého artefaktu zahrňuje jeho význam, prípadne spôsob manipulácie s ním. V závere teoretickej časti je ešte vysvetlený nástroj nazývaný RACI matrix, teda matica zodpovedností, ktorá je v práci použitá na mapovanie a vizualizáciu úloh vykonávaných v rámci vývoja produktu a typom zodpovedností, ktoré k nim majú jednotlivé role projektu.

Druhou a rozsahovo najväčšou časťou tejto práce je kapitola zaoberajúca sa Analýzou súčasnej situácie. V úvode tejto časti je predstavená firma a jej organizačná štruktúra. Ďalej nasleduje popis projektu, ktorý je uvedený vyššie v rámci tohto abstraktu. Po tom nasleduje detailný popis procesu vývoja produktu, ktorý sa skladá z viacerých fáz, v ktorých má každý člen tímu určené svoje úlohy a zodpovednosti. Analýza ďalej mapuje role, udalosti a artefakty procesu. Dôkladne sú tu prebrané ich vzájomne asociácie, účel, zodpovednosti a podobne. Výstupom analýzy je zoznam nedostatkov a odlišností aktuálne používanej metodiky od metodiky SCRUM. Konkrétne analýza odhalila to, že osoby zastupujúce rolu Scrum Mastra vykonávajú technické úlohy určené pre vývojárov a tým pádom zanedbávajú podstatu svojej role, ktorou je aplikácia a dohľad nad dodržiavaním metodiky SCRUM. U role Product Owner bolo v rámci analýzy zistené, že správne vykonáva aktivity spadajúce do správy artefaktov Product Backlog a Sprint Backlog. Táto rola však nesprávne vykonáva aktivity ako tvorba návrhov riešení a testovacích scenárov, analýza požiadaviek na novú funkcionalitu, technická podpora vývojárov, priraďovanie úloh konkrétnym vývojárom, monitorovanie výsledkov testovania, tvorba dokumentácie správa automatizovaných testov a iné. U role Developer je v rámci súčasného stavu projektu chýbajúca prítomnosť samo-organizovanosti. Pozícia Systémového Architekta je z kontextu aplikácie metodiky SCRUM vyhodnotená ako zlučiteľná s rolou Developer. Čo sa týka udalostí vývojového procesu, Scrum bol vyhodnotený ako dobre štruktúrovaný. Sprint Planing bol vyhodnotený ako nedostačujúci v otázkach definície obsahu Sprint Backlog ako aj procesu tvorby obsahu Sprint Backlog, ktorý by mal byť vytvorený rolami Developer prostredníctvom diskusie so zamestnancami v role Product Owner. Takisto by mal byť kladený väčší dôraz na formuláciu Definiton of Done. Daily Scrum nemá v súčasnom stave urečenú osobu moderátora, ktorý by dohliadal na dodržiavanie jeho časového, ale aj obsahového rozsahu. Sprint Review by sa mal správne konať na konci Sprint a mal by obsahovať vyhodnotenie splnenia úloh z Sprint Backlog a nie iba prezentáciu dodanej práce, ako je to v aktuálnom stave. Sprint Retrospective by mal navštevovať aj Product Owner a malo by byť vynaložené väčšie úsilie na odstránenie nedostatkov, ktoré táto udalosť odhalí. Analýza tiež odhalila, že Product Backlog a Sprint Backlog by potrebovali viac organizovanosti a pravidiel v rámci manipulácie s nimi.

Návrhová časť obsahuje návrh na odstránenie uvedených nedostatkov. Tímová štruktúra je rozšírená o dvoch nových členov, ktorými budú zamestnanci zastupujúci rolu Scrum Master. Zamestnanci zastávajúci túto rolu v aktuálnom stave projektu budú presunutí na role Team Leader, kde môžu využiť svoje technické znalosti a znalosť produktu a zároveň budú odbremenení od úloh zameraných na aplikáciu metodiky SCRUM. Dvaja noví členovia v role Scrum Master si rozdelia tímu tak, že každý bude mať pod správou dva tímy, v ktorých bude dohliadať na dodržiavanie princípov SCRUM. Táto rola je v rámci návrhu kompletne zbavená zodpovednosti za vykonávanie úloh patriacich do vývoja. Rola Product Owner je taktiež zbavená vývojových aktivít a bude sa plne zameriavať na maximalizáciu hodnoty produktu a transformáciu zákazníckych požiadaviek do vývoja. Vývojári označovaní názvom Developer sa budú aktívnejšie zapájať do procesu plánovania a ďalej budú vykonávať úlohy patriace do Sprint Backlog a Product Backlog. Rola Team Leader rozširuje rolu Developer o právomoc rozhodovania o technických riešeniach v potrebných situáciách a pridáva jej taktiež zodpovednosť za úlohy priradené tímu ako aj povinnosť poskytovania podpory menej skúseným členom tímu. U udalostí sú v rámci návrhu odstránené nedostatky uvedené v analýze, retrospektíva je umiestnená na koniec Sprint a zároveň je pre ňu navrhnutá nová, interaktívnejšia forma. Všetky udalosti budú moderované rolou Scrum Master.

Práca obsahuje aj finančné ohodnotenie navrhovaných zmien, ktoré sa skladá z dvoch častí. Prvá časť sa zaoberá zvýšením fixných nákladov spoločnosti vyplývajúcich z existencie dvoch nových členov projektového tímu. Odhad navýšenia fixných nákladov bol vyčíslený na sumu 2 970 960 Kč ročne. Druhá časť nákladov má jednorazový charakter a týka sa vzdelania a certifikácie členov projektu v metodike SCRUM. Odhad tejto časti nákladov bol realizovaný v troch variantoch. Prvý variant spočíva v inertnom školení zamestnancov, ktorí nedisponujú oficiálnou certifikáciou, ktorý bol ocenený na sumu 306 068 Kč. Druhý variant predstavuje kompromis v rámci šetrenia finančných zdrojov a obstarávaním certifikácie zamestnancov. Oficiálneho školenia sa v tomto prípade účastina iba zamestnanci, ktorí v aktuálnom stave nemajú certifikáciu. Tento variant predstavuje náklady vo výške 817 600 Kč a po jeho aplikácií by mali všetci členovia projektu oficiálnu certifikáciu. Posledný, najdrahší variant, spočíva v preškolení všetkých členov projektového tímu prostredníctvom oficiálneho školenia. Po ukončení školenia by mali všetci členovia projektového tímu najvyššiu možnú certifikáciu odpovedajúcu ich SCRUM role, ktorá je dosiahnuteľná v rámci školení poskytovaných v Českej Republike. Tento variant predstavuje jednorazový výdaj vo výške 1 086 068 Kč.

Záverečná časť návrhu patrí zoznamu prínosov navrhovaného riešenia, ktoré sú lepšia reakcia na požiadavky zákazníkov a ich transformácia do produktu, presnejší a efektívnejší proces plánovania, zlepšenie kvality vyvíjaného produktu, upevnenie vzťahu so zákazníkmi, zefektívnenie pracovného procesu, priaznivejšie prostredie pre rast členov projektového tímu a zvýšenie efektivity vývojového procesu. Všetky uvedené benefity by sa mali priaznivo odzrkadliť na finančných profitoch projektu.

Declaration

I declare that the submitted Master's thesis is original and I processed it independently. The quote of the bibliography is complete, and I did not violate any copyrights (in terms of Act no. 121/2000 Coll., about Copyright and right related to Copyright).

. . . .
Matúš Burzala
May 8, 2021

Acknowledgements

I would like to express my sincere gratitude to my thesis supervisor Ing. Lenka Smolíková, Ph.D. for her valuable guidance and advice that she has provided throughout the writing process of the thesis. Also, I would like to say thank you to all my friends and family members that gave me support during whole my studies.

Contents

Introduction **3**

Goals of thesis and used methods **4**

1 Theoretical review of a problem **5**

1.1 Software development life cycle . . . 5
1.2 Traditional models of software development life cycle . . . 8
1.2.1 Waterfall Model . . . 8
1.2.2 Iterative Model . . . 9
1.2.3 Spiral Model . . . 10
1.2.4 V-Model . . . 12
1.3 Agile models of software development life cycle . . . 14
1.3.1 Crystal . . . 15
1.3.2 Extreme Programming . . . 16
1.3.3 Lean Software Development . . . 18
1.3.4 Kanban . . . 20
1.3.5 Scrum . . . 21
1.4 RACI matrix . . . 28

2 Analysis of contemporary situation **30**

2.1 Company Introduction . . . 30
2.2 Project introduction . . . 32
2.3 Process of the product development . . . 35
2.3.1 Change request . . . 36
2.3.2 Analysis of the change requests . . . 37
2.3.3 Solution design . . . 39
2.3.4 Implementation . . . 41
2.3.5 Testing . . . 42
2.3.6 Deployment . . . 43
2.4 Roles and responsibilities of the process . . . 43
2.4.1 Scrum Master . . . 43
2.4.2 Product Owner . . . 45
2.4.3 Developers . . . 46
2.4.4 System Architect . . . 46
2.4.5 Responsibilities within the process . . . 46

2.5 Events and artifact of the process 47
2.5.1 Sprint 47
2.5.2 Sprint Planning 47
2.5.3 Daily Scrum 49
2.5.4 Sprint Review 49
2.5.5 Sprint Retrospective 50
2.5.6 Product Backlog 50
2.5.7 Sprint Backlog 50
2.6 Analysis summary 51

3 Proposal of solution **57**
3.1 Proposal of changes 57
3.1.1 Team structure 57
3.1.2 Scrum Master 59
3.1.3 Product Owner 59
3.1.4 Developers 61
3.1.5 Team Leader 62
3.1.6 System Architect 63
3.1.7 Responsibilities within the process 63
3.1.8 Sprint 65
3.1.9 Sprint Planning 65
3.1.10 Daily Scrum 66
3.1.11 Sprint Review 67
3.1.12 Sprint Retrospective 67
3.1.13 Product Backlog 68
3.1.14 Sprint Backlog 69
3.2 Financial evaluation of the proposal 69
3.3 Benefits of the proposed solution 74

Conclusion **76**

Bibliography **78**

Appendices **80**

Introduction

In today's age of digitization and rapid technical evolution, technologies are experiencing extreme, or even exponential development. This fact is affecting costumers' behaviour and therefore setup of the whole global market. What is today perceived as a cutting-edge technology or effective tool with modern design, can be tomorrow labelled as an old-fashion piece with an outdated look with a deprecated functionality. This attitude is naturally reflected in the customers' requirements towards the products regardless of the market sector. Producers are forced to answer to these changes and must put enormous effort into the fulfilment of the customers' needs in order to defend their spot on the market, keep pace with the competitors, and hold customers' loyalty.

This is the time when Agile methods are coming on the stage. Thanks to their focus, which is placed on the adaptability to the customers' requirements, they become more and more popular not only in the informatics field. Thanks to their key values which are the promotion of individuals and interactions over the processes and tools, creation of working software over comprehensive documentation, customers collaboration over contract negotiation, and response to changes over following of the plan, their application on the projects provides a fertile environment for the creation of successful products.

This diploma thesis deals with the application of the agile methodology SCRUM on a software development project. The first chapter of the thesis 1 contains theoretical information that is important for proper understanding of other parts of the thesis. The second chapter 2 focuses on the analysis of the contemporary situation which includes the introduction of the company and project, a detailed description of the development process on which SCRUM methodology will be applied, and summarization of roles, events, and artifacts that are existing in the current state of the project. The analysis is closed by summarization of all differences that the current state has comparing to the definition of SCRUM. The last part of the thesis 3 contains the proposal of the solution which is based on the information gathered in two previous chapters. The proposal contains a list of changes in the project's team structure, responsibilities of the roles, a form of development process's events, and adjustments of artifacts used in the process. The proposed solution is also evaluated from the financial point of view and the whole proposal is terminated by the summarization of the expected benefits, that should SCRUM implementation bring up into the project.

Goals of thesis and used methods

The main goal of the thesis is an application of SCRUM methodology on a software development project in order to optimize the working process. Considering the fact that the methodology that is currently used on the project is based on SCRUM, specific characteristics of the product and its development process, and the organizational structure of the company, this main goal incorporates several complementary goals.

One of the complementary objectives is an analysis of the contemporary situation on the project which includes a detailed analysis of the process of the product development. Another goal of the analysis is the identification of the roles that are participating in the development process and summarization of their responsibilities and activities in the development process as well as towards other participants of the process. The analysis also needs to gather information about the artifacts and events of the process. All outcomes of the analysis will be used for comparison with the SCRUM definition in order to find point out differences, which removal should bring more efficiency into the process.

Another complementary objective towards the goal of an optimization of the development process is a proposal of the changes that will lead to proper implementation of the SCRUM methodology. Actions listed within the proposal need to respect the character of the product its development process and the organizational structure of the company. The focus needs to be put on the specification of the team structure and its roles. The needed outcome of the proposal is also a formulation of the events and schedule of the development process. Also, a list of artifacts used in the process needs to be provided with the detailed specification of their agenda, participants, and accountabilities of the participants. It is important to provide a financial evaluation of the proposed solution in ordered to provide direct arguments for easier decision making about the provided solution.

The thesis utilizes methods and techniques for gathering information such as direct observation with active participation in the observed process as well as interviews. For better visualization and clarity of analysed facts, there is the presence of the EPC diagram and RACI matrixes within the thesis.

Chapter 1

Theoretical review of a problem

This chapter contains theoretical information, that is necessary to know in order to properly understand other analytical and practical parts of the thesis. Most of the information, except content that has its resource explicitly listed in itself, outcomes from resources [10], [3], [12], [13], [4], [15] and [17]. The first section 1.1 of the chapter contains a brief introduction to the Software development life cycle followed by a short description of its phases. The second section 1.2 sequentially describes four well-known modes of Traditional software development life cycle. Descriptions of each of the four selected modes are briefly explained basic principles of the modes followed by the model's diagram and list of advantages and disadvantages. The third section 1.3 discusses the Agile models of software development. At the beginning of this section, the first representative of agile models Crystal 1.3.1 is introduced. Afterward, Extreme Programming 1.3.2, Lean Software Development 1.3.3, and Kanban 1.3.4 are depicted. The whole chapter is concluded by a detailed description of the Scrum method 1.3.5, its roles, events, and artifacts. In the very last section 1.4, there is a short paragraph describing the RACI matrix, its purpose, and structure.

1.1 Software development life cycle

Software Development Life Cycle (SDLC) is a methodology used in the software industry in order to create high-quality software products. SDCL defines the process, that aims at the high quality of the developed software, which is following customer's requirements, and which is delivered within the intended budget and time. All these aspects are achieved by following a well-structured plan that consists of six phases 1.1 – Requirement Analysis and Planning, Requirement Definition, Architectural Design, Development, Testing and Deployment, and Maintenance.

Requirement Analysis and Planning

The essence of this phase is to obtain input from all stakeholders which are customers, sales department, marketing department industry experts, and so on. The result of this phase should be the answer to the question of what problem our project should

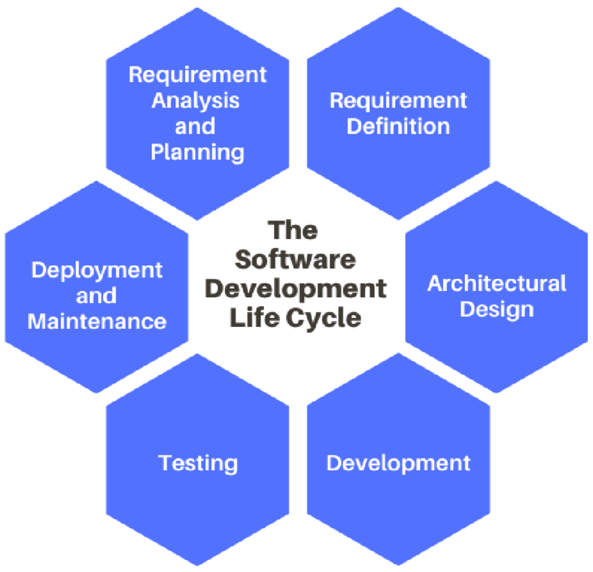
Requirement Analysis and Planning
Requirement Definition
The Software Development Life Cycle
Deployment and Maintenance
Architectural Design
Testing
Development

Figure 1.1: **The Software Development Life Cycle and its phases.** (Source: Own Creation)

solve, what functionality needs to be delivered or what needs to be improved on the product. The output of this phase is a feasibility study from the technical, technological, economic point of view. The feasibility study should include various approaches that should lead to the successful implementation of the project.

Requirement Definition

In the second stage, it is important to clarify all product requirements. Afterward gathered requirements must be approved either by the customer, market analyst, or another responsible person. For these purposes, a document named Software Requirement Specification (SRS) is used. SRS contains all product requirements that must be fulfilled during the project life cycle.

Architectural Design

In this phase, SRS which is base for software architects is transformed into Design Document Specification (DDS). This document usually contains more than the one suggested design approach for the document architecture. A key part of this stage is a review of the DDS by all stakeholders who are considering all aspects such as modularity, robustness, scalability, risk evaluation, financial, technological, and time constraints. As an output of this review, the best approach is selected for product development. The selected design includes all technical details necessary for the implementation of the required product or module, as well as a description of its interface required for future integration or deployment in the customer's environment.

Development

At this stage, the actual product development begins. Developers involved in the creation of the product's source code must adhere to the design contained in the DDS, as well as the established standards and procedures used within their organization. The time spent on development depends on the amount of knowledge and experience of development team members, as well as on the quality and sophistication of the DDS created in the previous phase. Technologies and tools used within the development process such as programming language, type of compiler, virtualization environment, and database engine are selected considering the character of the developed product.

Testing

The goal of the testing stage is to reveal all the shortcomings and defects of the software. Founded errors are repeatedly reported, corrected, and retested until the product reaches quality standards defined in the SRS.

Deployment and Maintenance

The last but not the least phase of SDLC includes two main activities, which are deployment and maintenance. When the product reaches the required quality, it is put into real operation. Deployment can be performed in many ways. Sometimes the product is initially deployed only among a narrower group of customers, for whom feedback is then collected. After correcting the product based on customer feedback, the product is widespread to all customers. This approach achieves a lower error rate of the product at the moment of delivery to the customer, but the release of the product is carried out later. In other cases, the product is spread directly placed on the market and is subsequently modified during operation in the form of new versions. The result of this is earlier delivery, which can, however, mean more errors in the initial versions of the product. Both approaches require maintenance and development of new versions throughout the whole life of the software.

1.2 Traditional models of software development life cycle

1.2.1 Waterfall Model

This model, illustrated on figure 1.2, was the first SDLC model used in software engineering. The process of software development is divided into several separate phases. Each phase begins whenever the previous phase ended and phases in this model never overlap. The output of the previous phase is used as an input for the following phase. Thus, this approach models the process of software development as a linear sequence of stages starting with the stage of Requirement Analysis followed by System Design, Implementation, Testing, Deployment and ending by Maintenance stage.

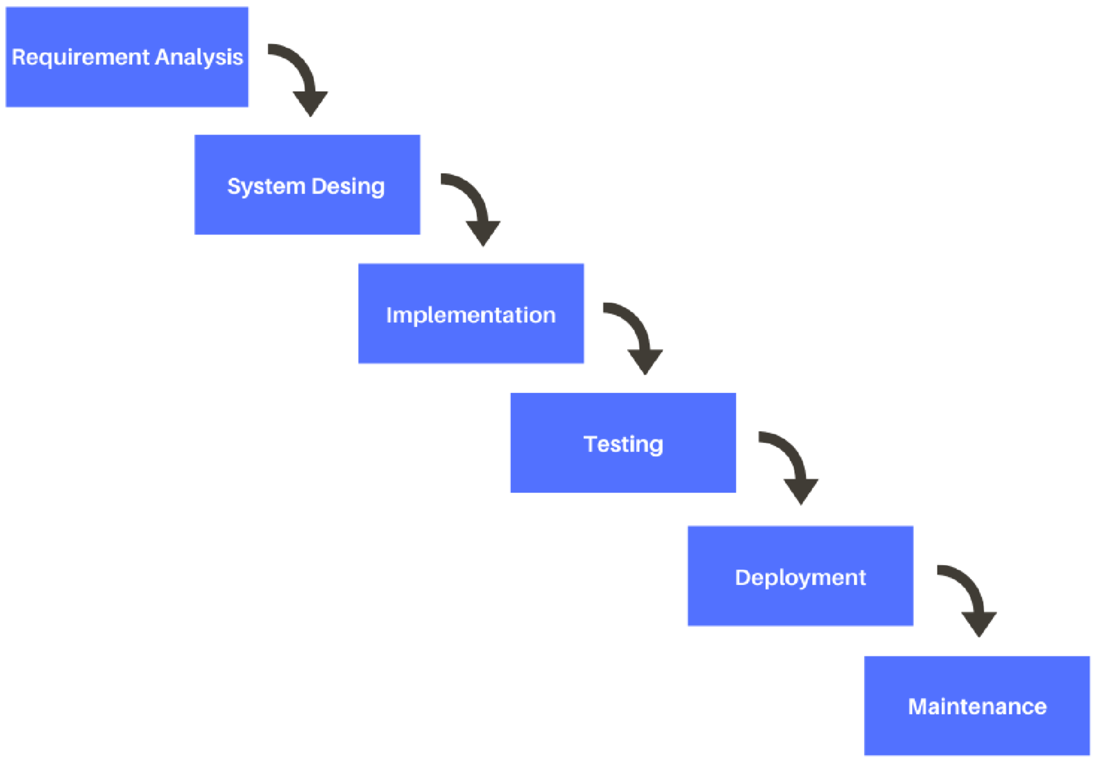

Figure 1.2: **The Waterfall model of SDLC.** (Source: Own Creation)

Nevertheless, this approach was invited a long time ago, it still very usable for a particular type of project nowadays. It is appropriate to apply this model for a project which has clearly documented and fixed requirements, product definition is stable, a project is quite short, and used technology is well understood.

Advantages

- Simple to understand and easy to apply
- Clearly defined structure and processes within stages
- Easy to manage, setup milestones and divide tasks

Disadvantages

- Difficult to reflect changes of requirements that appear during the process of development
- Functional product is available at a later stage of the project
- Errors made in the initial stages are difficult to eliminate in the later stages
- In case of change of requirements whole process must start from the beginning which is reflected in costs or can cause failure of the project

1.2.2 Iterative Model

The essence of the iterative model 1.3 is to develop software through repeated iterations. It starts with the implementation of a subset of requirements. In every further iteration, results achieved in the previous iteration are reviewed, and the list of requirements is updated. Then, new requirements are transformed into new versions. Each iteration consists of requirement specification, design, implementation, and testing phase. This cycle is repeated, and new versions are produced and repeatedly adjusted or enhanced until the complete system is developed and prepared for deployment.

This model fits projects that have clearly defined requirements for the complete system. As is mentioned in the description of the model above, some adjustments to the system can be performed during the development, but it is necessary to mention that major requirements must be defined at the beginning. Another use case of this model is when the development team uses new technology, which it is learning during the process of development.

Advantages

- Functional software (with limited functionality) is available in early stage of the project
- Parallel development and work in different phases are possible
- Results are obtained periodically and therefore can be reviewed constantly
- Possibility to change requirements during production process
- Less costs in case of changes in customer’s requirements for the system

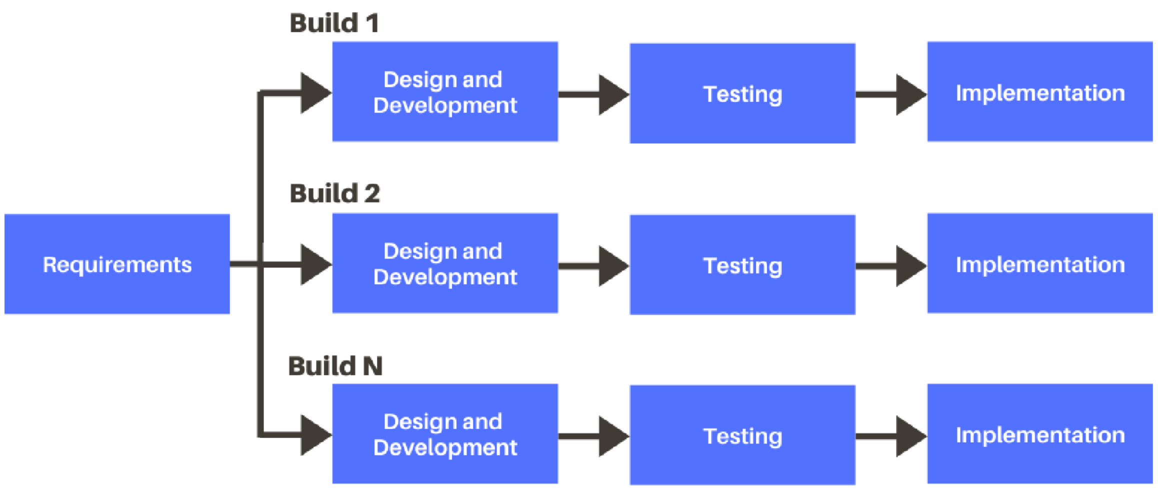

Figure 1.3: **The Iterative model of SDLC.** (Source: Own Creation)

Disadvantages

- More difficult to manage whole process
- Significant changes of the requirements are not easy to apply
- Weaknesses or issues of system architecture can pop up during later iterations of the development with incoming requirements

1.2.3 Spiral Model

This model incorporates evolving approach from the Iterative model combined with the stepwise approach of the Waterfall model. A big emphasis is placed on the support of risk handling. The Diagram of the model 1.4 looks like a spiral divided into four quadrants. Each quadrant represents one phase of the development process and during the development project repeatedly rotates through stages. Each loop in a Spiral model is called the Phase of the software development process. Product prototype is provided at the end of each phase. A final number of phases is unknown at the begging of the project and can dynamically change. The role of the project manager has huge importance here in the determination of the number of phases.

First stage by which each phase of the spiral model start starts can be named Objectives determination. Here, customer's requirements are gathered and analyzed. Based on that, objectives are conducted, and alternative solutions are proposed. In the next stage, proposed solutions are evaluated according to the objectives and constraints in order to select the best option. The focus of the evaluation is on risk perception. I the end of this quadrant prototype of the best solution is built. The third stage, stage of development and testing, includes implementation of the source code followed by verification through testing. At the end of the stage, the new version of the product is available. The last quadrant is named Review and Planning. Here customer evaluates the current version of the software. If the current version still doesn't meet requirements plans for new iterations are prepared.

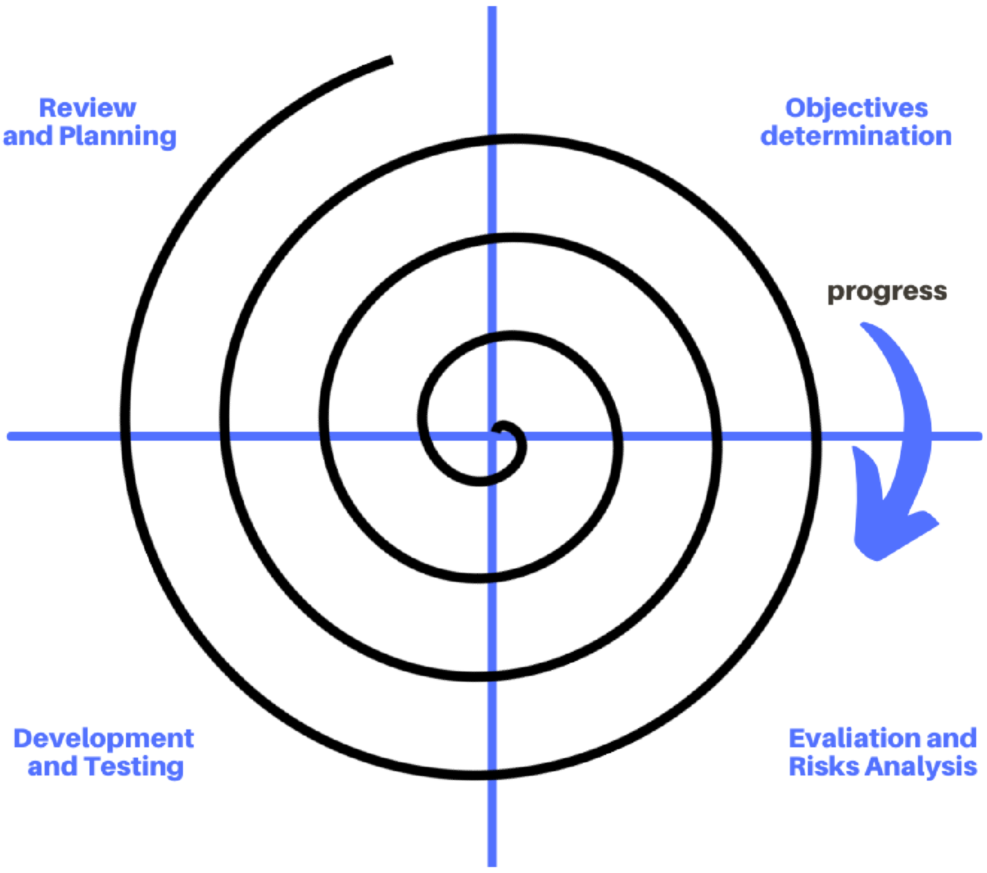

Figure 1.4: **The Spiral model of SDLC.** (Source: Own Creation)

This model mostly fits the large projects with a sufficient budget when it is expected that significant changes may be required in the future. Also, this approach works with systems that have unclear and complex requirements at the beginning of the project. Projects that need frequent releases of the new version for customer evaluation also benefits from this model.

Advantages

- Significant changes of requirements are acceptable

- Better risk management
- Relatively high number of versions for customer's evaluation
- Functional product in early phase of the project

Disadvantages

- Structure of the process is complex which means higher demands for management
- Hardly to estimate end of the project
- Risk analysis needs experienced specialists
- Not suitable for small projects

1.2.4 V-Model

V-model is also known as the Verification and Validation model can be considered as an extended version of the Waterfall model. In advance to the classical sequential linear approach in the Waterfall model for each stage of development (except the coding stage), there is one corresponding stage of Validation. As we can see on the diagram 1.5, the model's name comes from its 'V' shape. The left side of the diagram represents a group of Verification stages and the opposite, the right side represents Validation stages. Both sides are connected by stage of the Coding. As well as in other sequential approaches each phase can only start when the previous ended.

The phase of verification involves static analysis which is performed without execution of the source code. Business Requirement Analysis is the first stage of this group. It includes communication with the customer in order to understand its expectations and gather system requirements. Acceptance test design is conducted as well in this stage. In the second step, System Design, based on a list of the requirements system design is completed followed by a plan of system tests. The Architectural Design stage includes processes such as system design creation, interface design, data flow design, technology selection, and integration test preparation. This stage is also called the High-Level Design. The last stage of the Verification phase is Module Design. It is also referred to as the Low-Level Design and at this point, the system is breaking down into separate modules. The purpose and functionality of each module are detailly processed as well as interfaces and data structures of them. Based on that, a unit test plan is constructed.

The coding phase, as it is obvious from its name, covers the writing of the source code.

The phase of validation carries four stages and involves dynamic analysis which is done by execution of the source code. Here we start with the Unit Testing stage which follows the Coding stage and is complementary to the phase of Module Design. This testing is performed on the code level and it aims at the elimination of bugs in the implantation of atomic units as methods, classes, or modules. The next stage, the

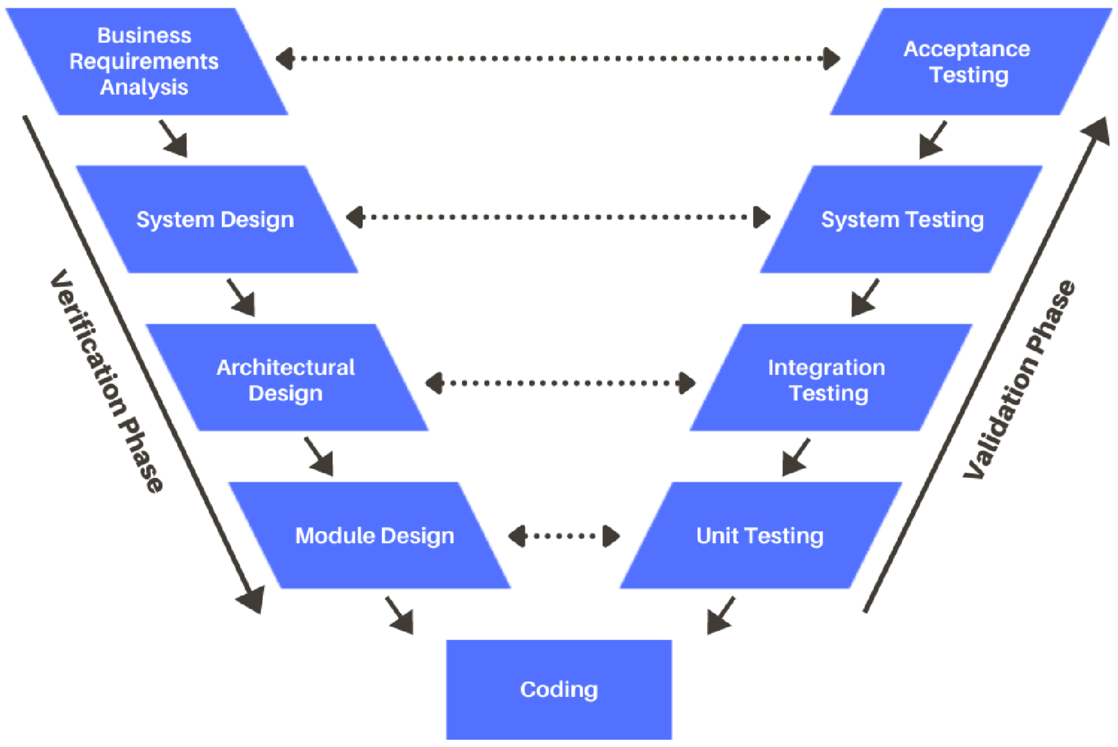

Figure 1.5: **The V-model of SDLC.** (Source: Own Creation)

Integration Testing stage performs testing focused on cooperation between modules. When particular modules are integrated into one system, we need to validate if they properly between each other and if their interfaces follow the High-Level Design. After all parts of the system cooperate as we want, it is time to test the system as one unit. For this purpose, V-model contains the System Testing stage. This checks the functionality of the whole system and its capability to interact with external systems. Acceptance Testing is the last step before releasing the products. It tests if the developed product fulfills business requirements and if it can work properly inside the customer environment. In this type of testing, we can also uncover hidden performance issues.

The application of this model is pretty much the same as in the case of the Waterfall model. In case of usage, system requirements should be very well defined without any expectations of further changes. Due to the huge emphasis on verification and validation of the developed system, this approach should be used for systems that requires high quality and maximal fault tolerance.

Advantages

- Easy to understand and use in practice

- Not difficult to manage, thanks to precious verification and validation system in each stage
- Huge emphasis on product testing

Disadvantages

- Zero flexibility in terms of changes in system requirements
- Existence of prototypes is missing
- Not suitable for huge or complex projects

1.3 Agile models of software development life cycle

Nowadays, requirements of the product functionality can change very dynamically due to newly invented technologies, behavior of end-users, trends in the market, or whatever external or internal factors. In the Agile process of software development, unlike in traditional models, the focus is placed mostly on adaptability to the current customer requirements. Customer interaction is one of the key characteristics of Agile modes. As it is defined in Agile Manifesto [13], the document that was conducted in 2001 by seventeen software engineers and which holds four values and twelve principles of agile software development, four values of Agile are:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

Agile development process starts with a definition of a set of features that the final product will comprise. This list of the features comes out from requirements analysis, and it is not followed by a detailed plan of tasks and milestones as we are used to, in the theory of traditional SDLC approaches. Agile breaks down SDLC into relatively short time frames during which the development team focuses on the creation of specific features. At the end of each time frame, a new build is released and presented to the customer and all other important stakeholders. Based on their evaluation, and with an aim on the rising business value of the product another feature is picked from the list and incorporated in the upcoming iteration. Thus, each iteration is incremental in matters of features, and the final build contains all demanded features. Atomic phases that each iteration consists of are Requirements and Planning, Development, Testing, Deployment, and Feedback.

Agile teams are supposed to be self-organized, self-motivated, cross-functional, and ideally located in one place. A self-organized and self-motivated team can deliver expected results in form of features, without the need for any external authority that would guide or encourage the team members on the way to reach settled goals. Co-location is another factor that improves a team's productivity. Cross-functionality can be described as the ability of team members to carry on all task that belongs to specific phases and roles within SDLC, besides that they are experts in their domains.

Thus, Agile models should be used on projects which require high flexibility and changeability in terms of product requirements. An environment where customers agree with a high volume of interaction whit the producer, or what is even better, demand frequent delivery, and continuous evaluation of the product is the right place to use these models.

1.3.1 Crystal

Crystal is a family of methods, that mainly focuses on people and the interaction while working on the project rather than on tools and processes. This family of lightweight and flexible approaches includes methods such as Crystal Clear, Crystal Yellow, Crystal Orange, and others. Crystal methods follow the idea that project characteristics change depending on the number of people involved, and also on the level of the criticality of the project. For example, a small project with few contributors does not require a huge amount of paperwork, communication, and reporting. On the contrary, large projects cannot be delivered without the need for a lot of frequent communication, status reporting, and paperwork. With consideration of that, the suitability of a particular Crystal method obeys three aspects – Team size, Criticality, and the Priority of the project.

As an example can be mentioned Crystal Clear method, which applies to projects with a team size of 1-6 members, a solid plan, and a fixed budget. As a comparison, a project that would involve 20-50 persons would likely require Crystal Orange. Such a project could last 1-2 years, the workforce would be split into teams according to their skills, releases would have an incremental character with a delivery period of 3-4 months, and so on. Other Crystal methods and their distribution according to the number of people involved are illustrated in the picture below 1.6.

All methods within the family are based on seven Crystal properties. First of all, Frequent delivery could be considered as key characteristics of all Agile methods, and which guarantee benefits such as earlier issue discovery, customer feedback, etc. Reflective Improvement aims at improving the product quality and techniques used by the team. This is reached by a discussion of current solutions and working processes. Osmotic Communications can be explained as absorption of the information or ideas by all co-located people, without intention to directly participate in the ongoing discussion. This can of course work only in small-size teams that are physically located in the same space. Personal Safety means trust in-between team members. This encourages them to stand up with their own ideas and opinions, which is essential for a correctly working and healthy team. Focus is a Crystal property that should developers have toward the tasks they are working on. It means they should have clearly defined tasks on which they can focus without any disturbance. Assurance of this boosts team performance and helps to follow deadlines. The penultimate

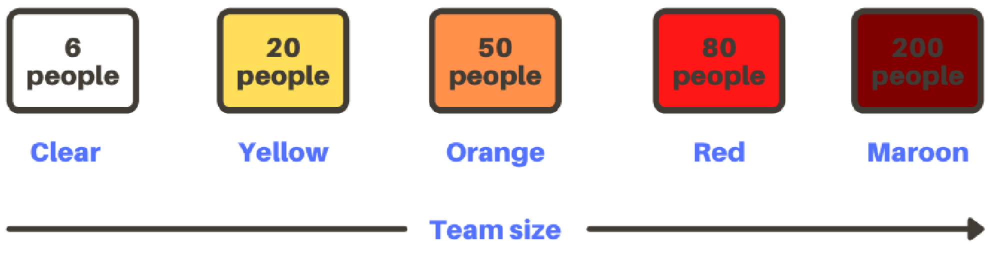

Figure 1.6: **Crystal methods and their distribution according to the number of people involved in the project.** (Source: Own Creation)

property is named Easy access to the expert users. This means direct communication with customers in order to get valuable feedback from real users. The last one, Technical environment and Frequent integration. This is a set of tools or machinery of automated testing and a continuous integration system that helps to find bugs in source code, and fix them in the early stages of the development.

Advantages

- Flexibility in terms of team size, project requirements and project or prioritization of delivery
- Due to the importance of teamwork and communication, team members can gain improvement of knowledge and can learn from each other
- Active participation of user and almost continuous feedback

Disadvantages

- Higher experience of team members is needed
- Need of co-location of team members due to proper communication
- Planning and development are not dependent on requirements

1.3.2 Extreme Programming

Extreme Programming is and software development methodology that is intended to produce high-quality software with high responsiveness to changing customer requirements. This is reached by frequent releases in short development cycles (with a span of 14 days' time frames) and checkpoints dedicated to obtaining new customer requirements. The name of the methodology comes from fact that all beneficial practices and activities used in traditional models of SDLC are here raised to extreme levels. Pair programming, the technique frequently used in Extreme programming, can be used as a convenient example of an extreme version of code review from SDLC. As code reviews are usually done in SDLC once upon a time (e.g., end of the specific period, end of the feature development), pair programming represents a session where two programmers are sitting next to each other while one is writing the code and simultaneously another from the pair is providing real-time code review.

Core representatives of Extreme programming practices that should be mentioned in this brief overview are - Pair Programming, Continuous Integration, Test-driven development, Planning Game, and Collective Ownership. Pair Programming is more detailly described in the paragraph above. Continuous Integration is an approach where contributing developers integrate the newly developed pieces of code to the one shared mainline several times per day. After the integration of the newly introduced source code, the whole system is tested. This supports the consistency of the product under development and helps to expose and fix integration issues sooner. And the fact that only a small part of the code is merged, makes the process of searching for the bug in the source code significantly shorter. Test-driven Development changes the order of phases during code writing and unit testing. Usually, at first, the production code is conducted and afterward unit tests are written and executed on the previously created code. But here, firstly unit tests are written based on new functionality requirements. The next step is the execution of the unit test and based on the failing tests, missing functionality is covered by coding. Planning Game is the name for the planning process of Extreme Programming. This process holds one meeting per iteration (usually one or two weeks) and has two parts. The first part called Release Planning is meeting with the presence of customers and developers. Here the definition of requirements for particular releases takes place. This part has three phases starting with Exploration Phase - where customer define product requirements which are transformed to the user stories (software system feature description in informal, natural language), Commitment Phase - were developers and business responsible commit in which release what functionality will be delivered, and ending with Steering Phase – withing which plan can be adjusted and list of product requirements is updated. The second part of the Planning Game is Iteration Planning. This has also three phases that holds the same names as phases in Release Planning. Only developers participate in this meeting. In the first phase, requirements are translated to the tasks, and tasks are written to the task cards. Within the Commitment Phase tasks are assigned to the developers and deadlines for the tasks are estimated. In the last phase, developers execute their tasks and match their results with the user story. Collective Ownership or team code ownership says that everyone is responsible for the whole code as well as everyone has the possibility to rewrite whatever part of the source code.

This methodology is suitable for projects with a high probability of frequent changes in product requirements. It can be used only when customers agree with participation in the project along with regular meetings and feedback sessions. From the producer's point of view, team members should be familiar with concepts as Pair Programming, Test-driven Development, and Continuous Integration. Also, these concepts need a specific development environment that contains appropriate tools and frameworks for unit testing, test automation, system integration, and so on.

Advantages

- Intense interaction with the customer and quick response to changes in requirements.
- Clean source code and frequent testing what guarantee high-quality standard.
- Importance of team collaboration and favorable environment for the growth of the abilities of team members.

Disadvantages

- Teams have to accept and follow specific techniques of the development process.
- The co-location of the team is needed in order to achieve efficiency.
- Due to customer's unclarity about the requirements, it is difficult to estimate the duration of the project as well as costs.

1.3.3 Lean Software Development

Lean method was initially developed for the manufacturing industry. Its principle is 'just in time production' and it aims at decreasing cost and increasing development process speed. The way how Lean reaches these goals is by reducing waste. Waste in terms of software development production are all activities or artifacts that don't add value to the product but consume effort, time, or financial sources. Particular seven wastes that appear in the software development industry are – Incomplete or partial work done, Additional or not required features, Extra processing, and documentation, Switching between tasks, Waiting periods and delays, Handing-off work between project members, and Product defects and bugs. For all these undesired wastes, Lean offers solutions.

All proposals and solutions for reducing software development waste, as well as the whole Lean philosophy, comes out from seven Lean principles 1.7 which are – Eliminate Waste, Empower the Team, Deliver it Fast, Optimize the Whole, Build in Quality, Defer Decisions and Amplify the Learning. In charge of fulfilling and following all these principles, there are three types of roles defined in Lean. First one, called Lean Master, a role that has to know and understand all Lean tools and techniques. With this knowledge, awareness of the project, and by keeping contact with customers, Lean Master is responsible for selecting project team members, coaching and mentoring them, and maintaining plan by considering all requirements and changes. Lean Project Leader can be perceived as a communication interface between Lean Master and Lean Team members. His usual tasks are leading the team and project, reporting to the superior roles, removing barriers, communication and mediation, and

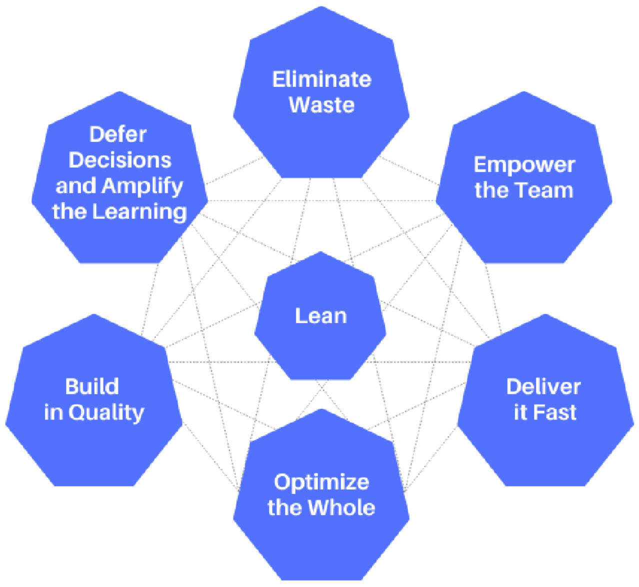

Figure 1.7: **Seven principles of Lean Software Development.** (Source: Own Creation)

responsibility for team improvement. Lean Team consists of developers, testers, and another domain expert depending on the character of the project.

Since Lean method requires the participation of the customer as all Agile methodologies, it can be used only for projects where customers are willing to do that. Another aspect that should be considered before choosing Lean is the size and complexity of the project, where Lean works only for smaller ones.

Advantages

- Reduction of costs and increment of development speed
- Focus on product quality and optimization of processes
- Amplification of team members' competencies and qualifications

Disadvantages

- Suitable only small and rudimentary projects
- High dependency of project success on team members

- Big demand of customer interaction

1.3.4 Kanban

Kanban is a Lean method that focuses on productivity and efficiency by managing the work flow with emphasis on the team's full capacity usage and development process optimization. The essence of Kanban is adjusting the amount of work in progress (WIP) to the team's capacity. It is also marked as a pull system because within Kanban work is pulled into the system when the team has free capacity for it, rather than tasks being assigned by superiors.

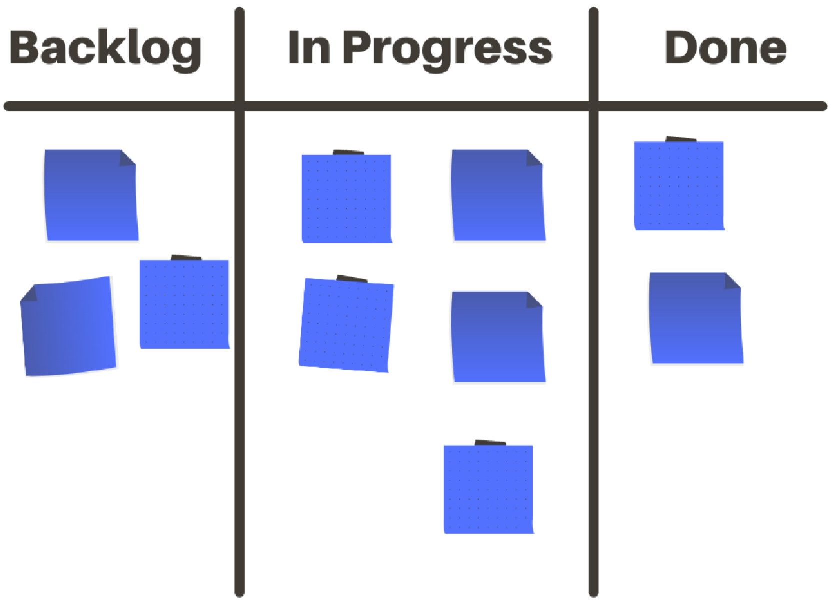

Figure 1.8: **Example of Kanban board.** (Source: Own Creation)

Kanban can be described over an explanation of its six core principles. The first one, Visualization of the Workflow, is the one that comes to mind of every person who ever got in touch with this method. The Kanban board, the crucial artifact of the method, provides transparent, easy to understand, and brief status of the work process. This board, usually in form of a whiteboard or virtual form, is divided into several columns. The most curricular columns, that every Kanban board (illustrated on figure 1.8) independently to the project are – Backlog, In Progress and Done. Each column is used for the determination of the task state. Tasks are represented by task cards. By moving the card from one column to another, team members are providing information about task status to all concerned. Only team members who work on the tasks can manipulate cards. Second, practice is a Limitation of the work in progress. Columns are not endless, and only a certain number of cards can be placed within them at once, which limits WIP and reflects the team's capacity. Different projects require a different number, capacity and types of columns (e.g. Waiting, Blocked, Verification) as well as distinct types of cards (Epic, User Story, Trouble Report). The third practice is called Mange flow, and it means that by using Kanban we are trying to achieve smooth workflow during the process of the development. Fourth in the list of Kanban practices is Making Process Policies Explicit. Application of this is done by keeping working process and all his aspects transparent, published, and accessible for all team members, because only then they can fully participate, increase value and bring positive impact. Feedback Loops are essential in Kanban, as Kanban itself comes from an agile family. A good example is the daily, max 15 minutes long, a meeting called Stand up. This meeting takes place in front of the Kanban board, and its agenda is that each team member share info about what he did the previous day and what he is going to do today, with the rest of the team. The last practice, Improve collaboratively refers to a way of teamwork that leads to better cooperation and easier direction to the improvement.

As this method is more philosophy and a way how to improve and raise efficiency, it can be implemented into projects which are fundamentally functional but could be smoother. Another place where Kanban could fit is in companies that are willing to improve exiting processes without radical system changes.

Advantages

- Strong visualization and transparency of the working process
- Easy to adopt into ongoing projects without the need for radical system changes
- Straight forward way to raise efficiency and productivity

Disadvantages

- Cannot be applied as a standalone tool, but rather merged with another process
- Not applicable for complex projects with a huge number of tasks and activities
- Prediction of dates for task delivery is difficult and inaccurate

1.3.5 Scrum

As The Scrum Guide [4] says: „Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems.“ This framework often referred to as a framework for IT development projects management, however can be applied to many different fields such as marketing, sales, research, etc. In its definition, it says that Scrum is simple, and it is definitely not a methodology. It is not a strict set of complex rules and practices on how to proceed. It provides an approach based on empiricism, that elects heuristic philosophy with support of human elements and self-organization rather than a heavy-handed algorithmic approach. Empiricism believes that knowledge comes from experience and observation.

Scrum framework models the work process as iterative and incremental, which goals into the optimization of predictability, and risk control. It defines three types of roles that play key roles within the main event of the Scrum called Sprint. Sprint is the name for one development iteration that is bound by other events, which will be described in further sections. These three roles and scrum events are supplemented by few artifacts that together represent the Scrum framework.

One of the three Scrum pillars is Transparency, which certificates that all work will be transparent and visible for all team members as well as project stakeholders. Without this, the empiric approach could miss-lead the project from its path towards increasing business value and customer satisfaction. The second pillar, Inspection, comes from flexibility and intention to uncover potential issues as soon as possible. It is processed by Scrum events and artifacts. The last pillar named Adaptation is closely interconnected with Inspection. After any deviation of the process is observed, an adaptation of the plan and task must be performed.

Scrum roles

Scrum recognizes three essential roles that together create a Scrum Team 1.9. Namely Scrum Master, Product Owner, and Development Team members. Besides these three roles, they are also stakeholders, which can be perceived as a customer that Scrum Team develops a product for. Scrum Teams are cross-functional what means, as one unit they have the skill set necessary to create value and fulfill the Product Goal. Another crucial characteristic is the ability to self-organization, which means that Scrum Team doesn't need an external authority that will decide who does what, when, and how. In terms of size, Scrum Team shouldn't consist of more than 10 people. The size of the team should be enough large to handle and finish significant work within one Sprint but at the same time enough small to stay agile. Here can be applied general rule, that it is easier to communicate and therefore be more productive among a small group of people. The scrum team takes responsibility for all product-related activities such as collaboration with stakeholders, planning, research, development, testing, deployment, maintenance, operation, etc.

Scrum Master

The Scrum Master is in charge of helping everyone within the organization and Scrum Team to understand Scrum theory and practice. He or she is responsible for establishing Scrum according to its definition in Scrum Guide.

The role of the Scrum Master is leader, not a manager, and his elemental duty is to empower Scrum Team's effectiveness. His or her main accountabilities toward the Scrum Team are coaching of the team members in cross-functionality and self-management, guiding of the team on the way of increasing product value by increments that meet the Definition of Done, removing barriers that distract the team from progress, facilitating and moderating of all Scrum events, ensuring that Scrum events take place at the right moment, with the participation of the correct participants, withing the defined timebox and they fulfill their purpose.

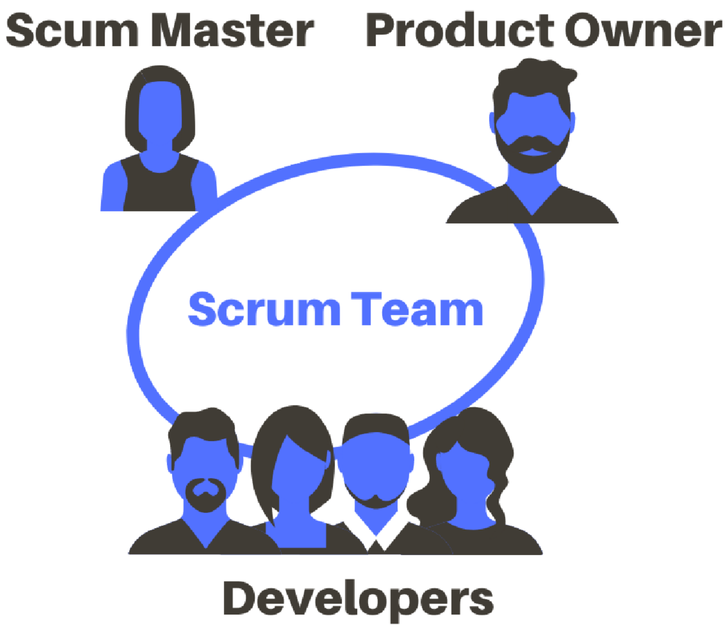

Figure 1.9: **Scrum Team.** (Source: Own Creation)

The services and activities that Scrum Master serves to the Product Owner can be summarized as assistance to find techniques for effective Product Backlog management and definition of the Product Goal, improving awareness of Scrum Team about the importance of clear and summary Product Backlog, facilitating collaboration between stakeholders, and support for the application of the empiric planning process.

The person in the role of Scrum Master has also his or her functions towards the organization which are: coaching, training, and leading the organization during the process of Scrum adoption, removing obstacles between the Scrum Team and stakeholders, helping all interested people to understand and achieve empirical approach for complex work, and proposing and planning Scrum application among the organization.

Product Owner

The main responsibilities of the Product Owner role are maximization of the product value and management of the Product Backlog. By maximization of the product value, it is meant an increment on the product after each Sprint, that is delivered by Scrum Team. Management of the Product Backlog consists of several activities beginning with the formulation and clear communication of the Product Goal, generation and straightforward communication of the Product Backlog and its items and ending with arranging of the Product Backlog items and keeping the whole Product Backlog transparent, accessible, visible, and properly understood.

Product Owner can delegate his or her accountabilities to someone from the Scrum Team or execute them by himself, but at the end of the day, he is always one that is responsible for it. The organization has to put trust in the Product Owner otherwise he or she can not succeed. His or her purpose is to represent the needs of stakeholders which he or she is transforming to the product by Product Backlog. Whoever wants change, can achieve it only by convincing the Product Owner.

Developers

Developers or better say Development Team, which consists of developers who are working on the tasks from Product Backlog, and who are providing increment of product value every Sprint. By the term developers, Scrum Guide doesn't mean only software developers, but all roles that taking part in product development such as architects, designers, researchers, analysts, programmers, testers, and many others according to the character of the product. The key qualities of the successful Development Team are, as was previously mentioned in other paragraphs, self-organization, and cross-functionality. Developers are always accountable for the creation of Sprint Backlog, preservation of quality by following the Definition of Done, everyday adaptation of the plan according to the Sprint goal, and mutual responsibility as professionals.

Scrum events

The cornerstone of all Scrum events (illustrated on figure 1.10) is Sprint. It determines when all other events take a place. According to the definition of Scrum, each event has a clearly defined timeframe, purpose, and list of participants. The reasons why all scrum events exist in their form are the formal opportunity to inspect and adapt Scrum artifacts as well as the provision of required transparency of the development process.

The Sprint

Sprint is a timeframe that encapsulates all other Scrum events, and it is time-space where all ideas are transformed into value. The duration of the Sprint is fixed, usually one month or less, and each sprint starts immediately after the previous one ended. With the increasing length of the Sprint, there is a higher possibility that the Sprint Goal becomes invalid, and the rate of complexity and risk might increase as well. Thus, by shorter Sprints, Scrum Team can benefit in the reduction of risk and with its associated costs. In the case of a Sprint goal become outdated, Sprint can be canceled. But it is necessary to say that only the Product Owner has such competence.

The sprint agenda consists of Sprint Planning, Daily Scrums, Sprint Review, and Sprint Retrospective, which are all actions that lead to the achievement of the Product Goal. By performing listed activities development process benefits in better predictability and adaption of the Product Goal. There are several rules that need to be followed during the Sprints, in concrete – it is prohibited to make any changes

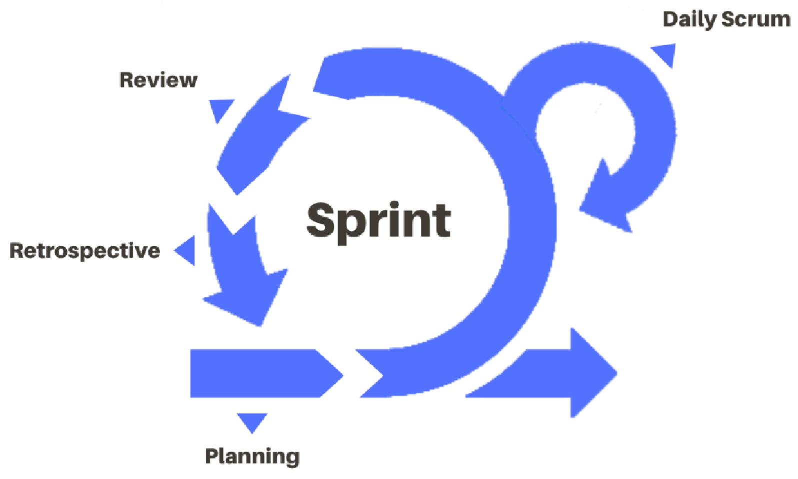

Figure 1.10: **Scrum events.** (Source: Own Creation)

that could end up in the threatening of the Sprint Goal, quality, in general, cannot be decreased, Product Backlog can be adjusted if the current situation requires it, and with the increased knowledge there is the possibility to update Scope by negotiation with the Product Owner.

Sprint Planning

Sprint Planning is an event that launches each Sprint. The outcome of this event is an artifact called the Sprint Backlog. Sprint Backlog consists of Sprint Goal, set of Product Backlog items selected for particular Sprint, and plan of the item's delivery. This output of the Sprint Planning is established by a collaboration of the whole Scrum Team. Scrum Team has also the possibility to invite other attendees to this event in order to receive their advice. The Product Owner has here responsibility to assure that participants are prepared to discuss an important item from the Product Backlog as well as the mapping to the Product Goal. In terms of the time duration of the Sprint Planning, for the Sprint that lasts one month, Planning is limited to a maximum of eight hours. I the case of shorter Sprints this event usually takes less time.

There are three topics that Sprint Planning has to resolve. The first one is 'Why is this Sprint valuable?' Here, the Product Owner is in charge to provide a proposal of product value increase for the current Sprint. Afterward, the whole Scrum Team hooks up in order to formalize a Sprint Goal. Sprint Goal must provide the answer why is this Sprint valuable to stakeholders.

The second topic focuses on ‘What can be Done this Sprint?’ The developers take items from the Product Backlog which will be included in the ongoing Sprint. This is done by a discussion with the Product Owner and if the situation requires it, refinement of the items in order to empower understanding and confidence can be done by the Scrum team. Estimation of the amount of work that should be delivered within the Sprint is no effortless activity. But how it comes from empiricism, with more experiences gained during previous sprints, the team becomes more accurate with their Sprint forecasts.

Third and the last topic that needs to be discussed during this event is ‘How will the chosen work get done?’ Developers step by step take each of the previously selected Product Backlog items and plan work that will result in an increment that meets the Definition of Done. At this point, it is very usual that items are decomposed into smaller tasks that should take one day or less. All this work is completely in hands of the Developers.

Daily Scrum

The main purpose of this short everyday meeting is to review progress on the way to reaching the Sprint goal. If it is necessary, adjustment of Sprint Backlog can be performed. Benefits of these events are impediments identification, communication improvement, quick decision-making promotion, and the abolishment of the necessity of other meetings. However, this event is not the only occasion when Developers can update their plan. They are free to meet whenever it is needed, to discuss more specifically about certain issues or about updating plans from the rest of the Sprint.

In order to decrease complexity Daily Scrum should occur every working day in the same place at the same time. It shouldn’t last more than fifteen minutes. The presence of the Developers is mandatory and in case that the Scrum Master and Product Owner are actively working on Sprint Backlog items, they are participating as well. There is no strict definition of the event structure, necessary is just fact of focusing on Sprint goal and formulation of a plan for the upcoming day. This builds focus and upgrades Scrum Team’s self-management ability.

Sprint Review

The Sprint Review is the penultimate event of the Sprint. During this event, the Scrum Team is presenting an output of the work executed by the time of the current Sprint to the important stakeholders. Accomplished progress towards the Sprint goal is discussed and determination of plan adjustments is carried out. According to the analysis, participants debate what to do next. Modification of the Product Backlog is also possible here with the aim to meet new opportunities. It supposes to be more a working session than just a passive presentation. Timeframe dedicated for such an event is four hours, for the one-month long Sprints. For shorter Sprints timeframe is naturally shorter.

Sprint Retrospective

As this is the event that is terminating the Sprint it is the right time for evaluation of previous decisions and processes. The Scrum Team looks at aspects such as individuals, processes, interactions, the Definition of Done, etc. Plan how to increase quality and effectiveness are discussed. Problems that appeared in previous weeks are explored, methods of their mitigation and solving are evaluated and solutions for removal of their origins are proposed. Proposals with the major impact are addressed and may be transferred into items of the Sprint Backlog. For the one-month long Sprint, it should take a maximum of four hours, and as usual, for shorter Sprints, the duration is shorter.

Scrum artifacts

The objective of the existence of the Scrum Artifacts is the maximization of transparency of principal information. Each artifact represents either work or value and each contains commitment. The intention of commitments is a transparent representation of the state with which real progress can be compared.

Product Backlog

The Product Backlog is the one and only source of work for the Scrum Team. It is an emergent and ordered list of required product improvements. Product Backlog items that Scrum Team is able to complete within one Sprint are prepared for the selection in a Sprint Planning. Backlog items are reaching this status by process of refinement. Refinement is the transformation of Product Backlog items into smaller and more rigorous items. This is a continuous activity that involves specification of the further details, providing and extending of the description, ordering, and size estimation. Developers who work on refinement activities are accountable for sizing while they are served by Product Owner assistance for better understanding and easier finding of compromises.

Commitment bundled with Product Backlog is named Product Goal. It is a long-term target for the Scrum Team. Before taking another one, they have to either fulfill the current one or abandon it. Product Goal represents the required state of the product which the Scrum Team's plans should lead to. It is inside the Product Backlog together with other items that specify which changes will fulfill the Product Goal.

Sprint Backlog

The Sprint Backlog is a plan constructed by Developers for themselves. It is a plan of delivering the Increment on the way follow Sprint Goal composed by a selected list of Product Backlog items. It is providing a transparent, highly visible, and real-time picture of the Developer's work for current Sprint. Its content can be updated during the Sprint with new knowledge learned. It should have enough explanation value that the Scrum Team can use for inspection of progress during Daily Scrums.

The Sprint Goal is all Developer's commitment to the Sprint Backlog. It is a single objective that Developers are aiming at while they are working on Sprint Backlog items. It is formulated during the Sprint Planning and afterward added into Sprint Backlog. All Developers should keep it in their minds throughout the Sprint, and if they feel their work is continuing by the different path that they expected, they are welcomed to negotiate with the Product Owner about the scope of the Sprint Backlog. This negotiation can conclude in the adjustment of the Sprint Backlog but can't affect the Spring Goal. This definite goal helps Scrum Team to work together on one common objective rather than on separate initiatives.

Increment

An Increment is a step forward on the way to the Product Goal. It must increase product value and work smoothly along with all previous Increments. Not all changes can be considered as Increment, only those which provide additional value by their usability. During the Sprint several increments can be incorporated into the product. Newly implemented ones are presented at the Sprint Review. However, the Sprint Review should not be the onliest point of Increment delivery. Some situations or stakeholders may request earlier delivery of demanded value. The rule that is used for evaluation of executed work, and which determines if provided work can be considered as Increment is called the Definition of Done.

Definition of Done is a detailed and formally formulated set of qualities or state of the product, which need to be entered in order to reckon it as an Increment. It is a tool used for sharing a transparent and understandable view of the work that was completed. If Product Backlog items don't meet this definition, they cannot be released or presented on the Sprint Review. In this case, they have to return to the Product Backlog where they will wait for their selection in further Sprint Planning. Definition of Done must be perceived as a standard within the organization or at least standard for the product. This standard represents the minimum that Developers must reach. Developers must subordinate to the Definition of Done and if there are multiple Scrum Teams cooperating on a single product, they must together define and follow the same Definition of Done.

1.4 RACI matrix

This subsection is based on information located in the source [9]. RACI matrix is a tool for responsibility assignment, which provides a simple and clear way how to map every task of the project and define which role has what specific type of responsibility towards the task. Usually, it has a form of a table in which rows are dedicated to the tasks of the process and columns are holding the information about the roles of the process. Each cell then represents the type of responsibility of a concrete role in a particular task. Acronym RACI stands for four types of responsibilities. Starting with the 'R' which means Responsible, and which is used for roles that work on the task and which complete the task. There can be several roles that are labelled with 'R'. Letter 'A' means Accountable, and the person to who belongs this letter is the one who approves that the task objective is completed. There should be only one Accountable person per task. ‘C’ stands for Consulted and people in this position are communicating with the Responsible during the task execution and their opinions are sought. The last letter ’I’ belongs to the roles that need updates about the task progress and these people are only informed about the task completion without contributing to the task.

Chapter 2

Analysis of contemporary situation

The following chapter contains an analysis of the contemporary situation of the observed process of software development. The purpose of the analysis is to map the development process and assign all activities that process incorporates to the particular roles which are executing them. The analysis also focuses on the investigation of events and artifacts of the process and their relation to the roles of the process. In concrete, the chapter starts with the introduction of the company 2.1, which includes a description of the organizational structure of the company. The next section 2.2 is dedicated to the project introduction, which explains what exactly is perceived as a project from the perspective of the previously introduced company. This brief description of the project illustrates what is the product and what activities are executed during the product development. Then, with the previously provided clarification of the product, the process of the product development is analysed. This is done within the longest section of this chapter 2.3. This section detailly describes the whole process of product development and decomposes it into several stages. Each stage is characterized in a separate subsection within which the ingress and egress of the stage are listed as well as activities that the stage incorporates. Activities are illustrated with the focus on roles that execute them. This section is followed by the section 2.4 which is introducing four different roles that appear in the process. The end of the section contains all responsibilities that specific roles have in the process. The penultimate section of this chapter 2.5 contains characteristics of five events and two artifacts of the process. The whole chapter is closed by section 2.6 that is summarizing provided analysis.

2.1 Company Introduction

Company XXX, which one of the projects is subject of this thesis, is the branch office of its mother-company YYY. The headquarters of the whole enterprise is located in the Nordic area. Besides branch XXX located in South Moravian city Brno, there are three more branch offices spread around the Czech Republic. XXX was established as a second one, and by the number of employees, it is also the second biggest among their sister's companies situated in the Czech Republic. The company

YYY is providing a variety of software solutions for financial, healthcare, automotive, energetic, retail, telecommunication, and many more industries.

In terms of numbers, the company employs around 24 000 professionals globally. Its annual turnover is estimated at approximately 3 billion euro and a major part of the revenue comes from Nordic countries. It provides its services to thousands of customers in more than 90 countries of the world. The history of the company began in the second half of the twentieth century. During many years of its existence, the company undertook several changes of the name as well as fusions with other enterprises. The branch located in Brno was found in 2015 and its most recent annual report indicates a turnover of 195 million Czech crowns and 135 employees.

Organizational structure

The simplified organizational structure of the whole corporate YYY is illustrated in figure 2.1. The root position within the chart is the CEO which is the highest position of the whole organization. If we look one step lower, we can see that organization is on this level divided into several areas. The scope of each area is determined by a combination of geographical location, field specification, or industry orientation. At this stage, the enterprise is a compound of more than ten parts from which each controls a different area and has its own head representative.

Each area is further divided into divisions. Branch office XXX represents one separate division. On the chart 2.1 borders of the division are marked by an orange dashed line. As it is illustrated, the whole Brno division has its own Program manager. The program manager covers the role of the division leader and serves the role of the contact point between division and corporate's top management. He is responsible for all strategic decisions of the division. Also, his task is to stand for the corporate values and to drive the division on the right path towards the corporate's business goals. Not each division has to follow the hierarchy applied in Brno, and substructures and roles of other divisions can diverse according to many factors.

Right below the Program manager, there are Line managers. As this thesis focuses on a concrete project, which has only one Line manager, information about a certain number of Line managers within the described division is not necessary. Also, there are other departments and external or internal positions that complete the division, such as Human resources, Accounting, etc. But since they are not taking part in the product development process, their presence in the chart would result in an unwanted increase of its complexity and decrease of its explanation value. The line manager of our particular project is supervising four teams. Each team consists of up to ten members. Team members have a wide scale of competencies such as development, testing, architecture design, continuous integration, computer networking, and so on. Level of experience and time spent on the project also diverse among the people. Each team sits together in a separate office, but some teams have members that work permanently remotely.

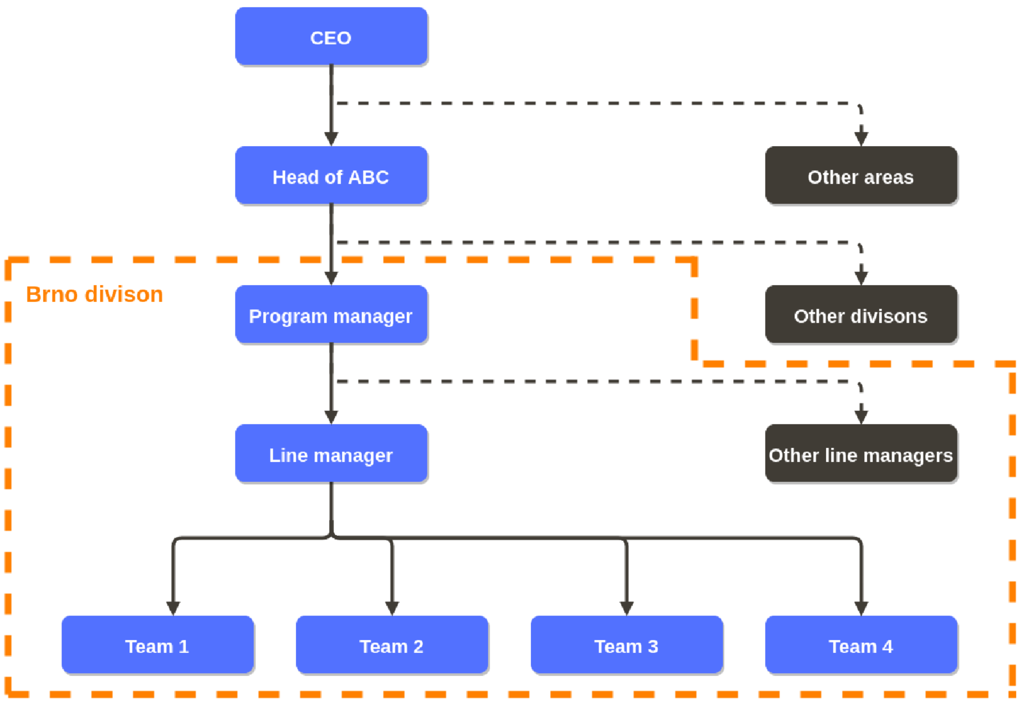

Figure 2.1: **Simplified organizational structure with focus on Brno division.** (Source: Own Creation)

2.2 Project introduction

In the beginning, it is important to clarify the definition of the project, and what is meant by the designation project within this thesis. The company YYY creates one product within several of its branch offices. Thanks to the size, complexity, and number of customers of the product, there are hundreds of people employed in the process of product creation and delivery. Each branch office, or in other words division, takes care of certain subsystem or logical module determined for example by the functionality. For instance, we can name divisions dedicated to the development of GUI (Graphical user interface), development of the hardware elements, development of module for user data network, a module for control system data, and others. Besides the divisions that are directly participating in the development process of the product, there are also divisions that are in charge of continuous integration system, acceptance testing, or technical documentation which is served to customers.

There is one separate group of people that is called the System council. Members of this group are professionals in several domains that have enough experience and competencies to officiate about the development of functionality of the whole product. This is the main authority that decides what functionality will be newly introduced to the product.

By the project (to which SCRUM will be applied), it is meant the process of the development of module for the user data network. In concrete, the bigger part of the module, because there are only four teams located in Brno out of six teams that are participating in the development of this module. Another two teams are completely separated from the teams in Brno, whatever we look at the physical location of the teams, management, position within the organizational structure, ownership of hardware devices used for development and testing as well as responsibilities for specific functionality of the module and its subsystems. The thing that bounds all six teams together is the shared source code of the module, communal responsibility for the module as one unit,and common System council.

Indeed, four teams located in the Brno division are taking part in the development of a single module. The source code of this module is shared among two other teams, which has however clearly defined parts of the module, which they own and take responsibility of. If the implementation of some functionality requires changes in the part of the code that belongs to teams outside the Brno division and vice versa, minor changes can perform someone from Brno's teams. In this kind of situation, it is usual that someone from the team, which is responsible for such code, provides code review and participates in the approval of the change. If a situation requires more significant changes, the request is created in form of a task that is addressed to the accountable team.

Also, the module must be compatible with other modules within the system and at the same time, it has to be consistent with the hardware components that it is orchestrating. Compatibility with other modules and hardware elements and therefore functionality of the whole product as one complete system is provided by Continuous Integration system, together with teams which are performing acceptance testing and simulation of real customer use cases. Currently, the product is already deployed among multiple customers. Therefore, another level of the testing and source of issue reports is provided by customers. Impulses for the creation of new functionality comes out from the System council, product users, customers, the internal necessity to broaden or improve a current functionality, following of market trends, business and economic suggestions, the existence of the new technologies, etc. Alike, a stimulus to redesign or extend current implementation often comes from the developers which have the closest distance from the project.

The development team along with the product source code provides also test coverage of the implemented code. Testing of the module as well as the whole product is performed by multiple stages. Each stage validates product quality from a different point of view and is performed at a certain moment. Starting with unit tests, the lowest level of the testing that verifies if particular methods and classes are providing outputs as they should as well as if they are able to deal with exceptions and incorrect user inputs. Unit tests are designed and implemented by each development team at the same time when the source code of the functionality is written. They are part of the product source code and their execution is triggered whenever the source code is being built. The second type of tests that are used within this project is named Integration tests. Word integration in their indication stands for the integration of the features between each other. A single feature can consist of several classes and module can carry a theoretically unlimited number of them. Thus, integration tests are the tool that checks if features inside the module can cooperate. They are part of the source code of each module as unit tests. Another difference, besides their purpose, is that they are segregated from other source code and they are executed by special scripts, which is triggered after the source code is pushed to the shared repository. The third level of the test is the system test, which verifies the functionality of the whole product. It means that they test all modules together along with the hardware components of the product. In some cases, hardware components are simulated by virtual machines. From the version control system and source code point of view, they are in an isolated repository. Within the testing process, they use hardware resources dedicated for these purposes with a combination of the reservation system for them. Besides that, they are using the external testing framework, an automated system for reporting the test results, and continuous integration machinery. All elements mentioned in the previous sentence are products delivered by other teams of the corporate or by external subcontractors. Source code of the tests is again developed by the development team. Execution of these tests is performed periodically. Their execution is orchestrated by the continuous integration system. If the number of new commits within the version control system reaches a specified number, repositories of all modules are built, bundled, deployed to the testing environment, and test cases are executed. Afterward, results are automatically reported and in case of the successful test results, a new version of the product is stored and forwarded to the last stage of testing. This test execution triggered by a number of commits operates only during the working days when developers are delivering their changes. Since the product is huge it would take a long time to test the whole functionality thus only the most critical parts are tested by this method. During the night-time and the weekends, the whole machinery provides the testing of a full scale of test cases, using the same procedure as daily testing. The fourth and the highest level of testing is acceptance testing. Provided by the external teams, it tests products by real test cases with more complex hardware topologies and by the presence of various software and network traffic.

Besides the development of the product functionality, an important part of the project is the creation of the technical documentation. There are several types of documentation included in this project. They diverse from each other by form, level of importance, and mainly by the addressee for which are wrought. The first type of documentation is documentation on the level of source code. This includes documentation of the methods and classes and comments inside the source code. There are no huge requirements for the form and quality of this documentation, but it has a key role in terms of product quality and sustainability. The second sort of documentation is API (Application Programming Interface) documentation of the module. It is generated by an automated tool from the source code, thus the creator must follow strict syntactic rules. This documentation is used mostly by other teams, which are using the module's API for communication with other modules. It is also used within the process of testing, whatever it is manual testing or creation of the automated tests. Authors of both previously mentioned types of documentation are members of the development teams. The last type of documentation, which is visible by the end-users of the product is customer documentation of the product. It detailly describes the functionality of all product features and offers instruction on how to configure, use, or troubleshoot the system. Because this is the source of the technical information provided directly to the customers, there is a big emphasis on the form and content. In an account of this, there is an external team, which based on the inputs provided by the development team members, creates documentation that follows all standards and policies.

2.3 Process of the product development

Process of the product development (shown on the EPC diagram 2.2) within the analysed project stands for the process of the development of module for the user data network. Several roles are taking part in this process, and their responsibilities and functions are described in the next section 2.4 of this chapter. The methodology used within this process comes from SCRUM.

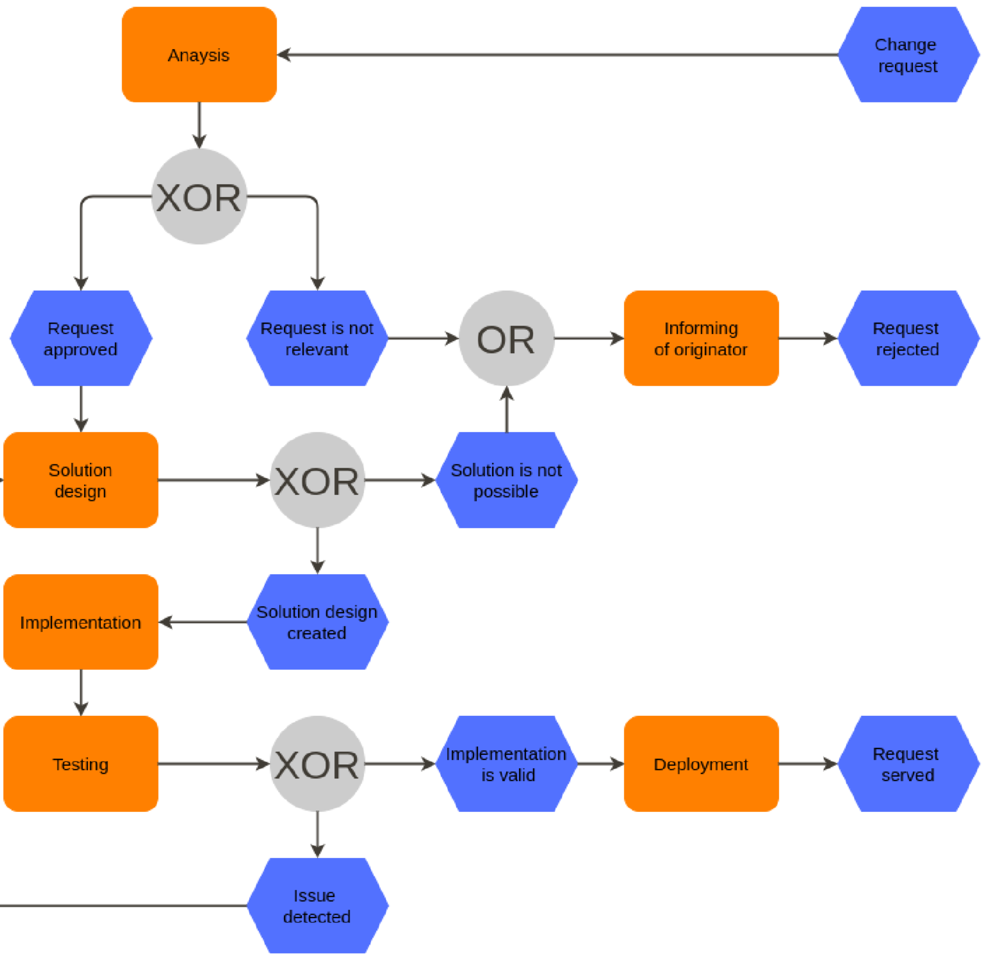

Figure 2.2: **EPC diagram of the process of product development.** (Source: Own Creation)

The whole process starts with the analysis of the change requests. There are multiple types of requests that are originating from different originators. According to the type of the request and its content, the request is analysed and then a decision of its approval or rejection is made. In case of rejection, this statement is provided to the originator of the request, and the existence of the request inside the process is fulfilled and terminated. If the conclusion of the analysis is the approval of the request, the next stop is solution design. This design is based on the information provided in the request and outcomes of the previously done analysis. If there is no possibility to transform the request to a thinkable design, the request is rejected. Again, the originator of the request is informed about this fact, and then the request considered as served. In the situation when the solution design ends up successfully, the process continues with the implementation. During the implementation phase, solution design is transformed into the source code. In order to verify newly created implementation as well as keep previous functionality in a consistent state, the development process incorporates a stage of testing. If there is any issue detected by the testing, a proposal of the solution and fix of the implementation is provided. If testing confirms the correct functionality of the implementation, the change can be deployed. Deployment within this process stands for the delivery of the change represented by the inclusion of the implementation into the shared project repository.

2.3.1 Change request

Inputs of the process of the product development are change requests. In concrete, for the observed process request for changes of module for the user data network. A change request can be the request for the creation of new functionality or modification of existing functionality. Within this project, requests are generated by different sources and have diverse forms and levels of detail.

The first type of request is requests for the new feature implementation. They come out from System council. The form of such request has is the text document, which has a strictly defined structure and detailly describes how, and what should feature provide. Creators of these documents are system architects. They obtain assignments from the SYSTEM COUNCIL within which they do research, in order to find out whether the implementation of the feature is possible, and how.

Another type of request that enters the process is a request for a customer's issue correction. There are dedicated Customer support teams, which are receiving requests from customers and transforming them into Trouble Report (further in the text TR) records inside the JIRA system (software for planning, tracking, and releasing of the software products [6]). Development teams then receive a report of the issue in form of the TR record within the JIRA. This record usually contains issue description, version of the customer's system deployment, characterization of the software and hardware configuration, and log records. Alternatively, a snapshot of the database can be attached to the TR. This kind of TR has always dedicated member of the Customer support teams, which is responsible for the TR and provides and further communication with the customer. Content of such TRs usually incorporates all necessary information. In case of any additional information is needed developers can ask using the comment section inside the TR record and the responsible support specialist will contact the customer. Customer's TR has generally the highest priority.

The third type of TRs is errors that come from testing. In case of an issue found on the stage of Acceptance testing, the team which is charring out such testing will create the TR record ad fulfill all needed data related to the issue. When the issue is observed on the level of System testing, there are two possible scenarios. Scenario number one, the issue appeared in the tests outside of the project scope. The finder of the issue must create a TR and forward it to the team responsible for the concrete area or to the Scrum Master, or Product Owner. Scenario number two, the issue shows up in test suites owned by teams from the observed project. Here finder of the issue can either remove the issue immediately or in case of a more complicated issue he can also create the TR. The possibility of immediate issue removal is thanks to the product complexity very rare.

The last type of change request is a request originating from the development team members. It can be for example proposal for code refactoring or redesign. The form of this request is usually a verbal proposition presented during the team meeting.

2.3.2 Analysis of the change requests

Each change request is analysed during the described process. The goal is an analysis of the request relevance from the perspective of added value for the customer, scope of the project, and current project priorities. It is also important to consider if the change fits into the product's concept and its functionality. The analysis is performed by the Product Owners, Scrum Masters as well as Developers. The possible results of the analysis are rejection or approval of the request followed by passing the request to the stage of solution design.

In the case of the new feature implementation request, there is no need for any extensive analysis. The necessary analysis was performed by the System architect yet before the delivery of the request to the development team. It is only needed to assign the feature development to one of the teams. This task belongs to the pair of Product Owners, which are aware of the team's free capacity. During the time of making a decision, they can discuss it with the Scrum Masters. But at the end of the day, the final decision is completely in their hands and the chosen team must accept the request. In practice, there are no situations when the Product Owner or the development teams refused such kind of request.

Another type of request that requires analysis is the customer's TRs. The fact that the concrete TR was delivered to the teams within the observed project is the result of Customer support team analysis. So, after the development team receives the customer's TR, the first question that needs to be answered is, which team is responsible for the particular area. This primordial analysis is done by the Product Owner. Firstly, he verifies if the issue belongs to the functionality of the module. Then he checks if the customer is really facing the product issue, or he just wrongly interpreted the product functionality. In case the issue is truly located within the module that the development teams take responsibility for, the Product owner assigns TR to the concrete team. In case of the issue comes from a different module, the Product Owner passes TR to the correct addressee. In case of the incorrect interpretation of the functionality, the Product Owner provides an answer to the originator of the TR.

In the response, he usually provides an explanation of the functionality along with a link to the sections of the Customer documentation which are describing particular functionality. The second step of the customer's TR analysis is a closes analysis of the product issue. After the Product Owner assigns TR to the development team, one of the Developers or the Scrum Master can assign the TR to himself and can start to analyse provided artifacts. Such analysis consists of searching for errors using log records, source code, a database snapshot, and comparing with the documentation. If it is necessary for analysis, the team member that performs the analysis can ask for further data using Customer support. The outcome of this second step is finding an error inside the product source code or inconsistency between database and hardware configuration. Also, the conclusion can be the incorrect interpretation of the functionality, which wasn't revealed by the Product Owner. In this team member has to follow the same procedure as during the first step. In case of inconsistency, the team should ideally search for the reason for the inconsistency and resolve it. Due to the time-consuming characteristic of such an approach, in practice, the team usually just resolves the inconsistency by a manual intervention of the database or the configuration. Thus, customer's TRs can be either rejected, in case there is no implementation issue. Also, they can be redirected to another, responsible team. Otherwise, customer's TRs can be accepted and rigorously analysed for purposes of solution design.

When it comes to TRs originating from testing, if the error was detected by the Acceptance test team, they create TR containing all necessary info and pass it to the Product Owner. Product Owner then continues as in the case of the customer's TR. If the issue was detected on the System testing level, a member of the development team accountable for monitoring the test suites does analysis. In the beginning, he verifies if the observed issue belongs to the scope that his team is responsible for. If the issue does not come from the team scope, he creates TR which then forwards to the Product Owner. Contrariwise, if the issue belongs to the team's scope, the developer has to continue with a more detailed analysis. If the observed issue has a trivial character, the developer should fix it. If the fix of the issue requires more time, the developer must create TR. In addition to data that is presented in all types of the TRs, he joins the results of his analysis. This kind of TR is then either passed to the Product Owner or if the developer has a free capacity, he can continue with the design of the solution. There is one special type of issue, that can be detected during the System testing. It is called an Environment issue. It can be an issue with the hardware in the testing lab, the error of the testing framework, or whatever issue that has no connection with the product. In that case, if it is a removable hardware issue, the developer should resolve it by himself. If it is a hardware issue that is not removable, the developer contacts the Lab support team. In case of any other Environment issues developer contacts teams that have responsibilities for those areas. Thus, requests in form of TRs originating from testing can be rejected, redirected to the other team, or accepted and analysed.

The goal of the request originating from the development team members is usually refactoring or redesign the existing implementation. Developers can come with requests whenever they see an opportunity. For instance, during the Planning, during other meetings, or they can address them in a private way directly to the Product Owner. Form of this kind of request usually texts or verbal. If the request represents less time-consuming activity, it can be approved by the Product Owner. In case of more significant change, which needs more time and more involved employees, it has to be approved by the System council. The main criteria used for evaluating the requests is an added value for the customer. Most of the refactoring and redesign aims on increasing of sustainability of the product source code. However, from the short-term point of view, this does not bring added value for the customer and therefore the majority of this kind of request is rejected. When the request was approved, the procedure is similar to the new feature implementation request. The only difference is that analysis is performed by the originator of the request, not by the system architect.

2.3.3 Solution design

Roles responsible for the creation of the solution design within the described process are Developers or Scrum Master. Solution design is based on information gathered during the analysis. At the same time, the design has to be in accordance with module architecture and product architecture which both have strongly established database and transaction mechanisms. During the process of the design creation its author consulates his design and obtains needed information from other team members, Product Owners, System architects, or other teams which superintend other modules. The final decision for approval of the design is in hands of the Product Owners. Currently, there are two employees in the position of Product Owner on this project, and both miss advanced experience in the domain of programming. This means that they judge design only from the perspective of functionality but not from an implementation point of view. This can result in a lack of focus on the quality of the implementation.

In process of solution design for the new features, the author is creating the design based on the document delivered within the change request. This document was prepared by the System architect, which summarizes all technical, technological, and also conceptual aspects. Thus, the document incorporates a description of the feature functionality, a list of selected technologies, tools, and protocols. Also, use cases of the feature cannot be missed in the document. Therefore, in pursuance of the described document, the product's source code, and the technical documentation of the product author build up the design. The design does not have predefined content. Usually, it consists of text description and graphical elements such as Entity Relationship Diagram, Class diagram, and so on. Design is stored on the project's Confluence web pages (workspace for sharing the knowledge [5]). The part of the design creation process is also the formation of the JIRA records such as User Stories and Tasks. This brings a need for the decomposition of the solution into multiple subtasks. The advantage of the decomposition is the possibility to divide work among more developers which should result in earlier delivery of the implementation as well as the better spreading of the knowledge about the feature around the team members. Besides that, it also makes planning and tracing of the progress easier. The aim is to formulate tasks in a way that their implementation should take a maximum of one working day. Achieving this is of course very difficult for example because of the different amount of knowledge and experiences of the team members across the domains. Within the delivery of the new feature, there is bundled delivery of the test coverage. Responsible for the test design can be either Developer which is designing the feature implementation, or another team member which has wider experiences with testing. The second option is more often taken into practice. Different level of the testing requires the different design. For Unit tests, the design is never provided. In the case of System tests and Integration tests, the design is conducted in form of the JIRA tasks. Task usually contains only text description. In the design of the System tests, there is one additional characteristic that needs to be established. This crucial attribute is the priority that is quantified according to the importance of the feature within the product. Based on the priority author of the design must choose a location where the new test will be placed in the Continuous integration system. His decision is affected also by the availability of hardware resources in testing labs. The majority of the attempt to create a solution design for the new feature ends as a success. This is caused mainly by the fact that they are based on the well-prepared analysis from the System architects. The architect usually reveals all technical and technological limitations. Thus, the only factor that is threatening design is the need for consensus with product architecture and real implementation.

Designing the solutions for customer's issues and issues originating from the testing is identical. These kinds of designs do not have a defined structure. The importance is put into the fastest delivery of the solution, which means that in practice team members only discuss the solution and afterward implement it into the source code. The absence of any solution documentation results in a non-systematic approach towards the product fixes. This affects the quality of the source code and at the same time possibility to reuse verified solutions by the other team members is disappearing. Usually, the developer prepares his private solution design in a form that fits him. He discusses this solution verbally with the Product Owner or other Developer. Afterward, he transforms the solution into the implementation without any further sharing of the solution among the communication canals of the team. In some cases, it is impossible to create a solution design. For example, when developers see a way how to fix the error, but by fixing such error other module functionality would be broken. For this kind of situation, there is a section 'Limitations and Workarounds' within the Customer documentation. When the developer comes to the conclusion that it is impossible to design implementation that would solve the issue, he documents a particular use case. Then he bundles the error case description along with the formulation of the manual procedure which resolves the error and joins it into the documentation. This Is afterward provided to the originator of the TR as a solution to his problem. If the developer designs solution for the customer's TR, he has to prepare a custom testing script. If the solution of the TR includes adjustments to the customer's configuration, it is important to prepare a proposal of manual testing scenario and a proposal of the deployment for the customer's environment.

When the team has to create a solution design for the request originating in the team itself, the originator of the request usually does it by himself. In case of complex change, the process of the design is similar to the designing of the new features. When the change is simple, the originator only creates a task or set of tasks that contain the formulation of the solution. According to the fact that this type of request is created when developers see the potential for improvement by the code refactoring or redesign, designing this kind of solution is always possible.

2.3.4 Implementation

The ingress of the implementation stage is solution design. During the implementation, solution design is transformed from the theoretical level directly into the source code of the module. Before the implementation, a particular developer picks the task, assigns its JIRA representation to himself, and sets the task status to 'In Progress'. This will tell all team members and other interested stakeholders what the task state is. In addition to the task status, everyone is allowed to comment on such a task using its JIRA record. Afterward, a developer can start with source code creation.

During the production, or modification of the source code should all developers follow commonly used patterns and best practices. The project has its own definition of code formatting, logging, and commenting. However, these definitions are not gathered in the same place and some of the developers don't know about them. In ordered to control adhering to the rules, maintaining the code quality, as well as verification the correctness of delivered changes, there is a code review system. This code review system is used around the whole product. For these purposes, the company uses the version control system Git (opensource version control system [14]). After creation or modification of the source code author of the change creates a so-called commit (record within the Git system, which represents and clearly indicates the change). Then, commit which contains also a text description of the change, and a link to the JIRA task is sent to the shared repository. There is one additional step after which the will be commit includes in the repository. This step is code review. For this purpose, the company uses a system named Gerrit (a system serving code review for Git repositories, which provides review, approval, commenting of the commits, and so on [8]). Commit appears in the Gerrit system after the developer posts them using Git. Here everyone with granted rights can see, review, approve or disapprove the commit and inform the author about certain irregularities using the comment section. Commit needs to be approved at least by one person except for his author, to be applied into the product repository. All developers are notified by email whenever someone posted a commit that contains changes to the source code, that they are authors of. In the ideal case, after receiving a notification should all developers provide the code review. Because as an author of the previous code they have sufficient knowledge and competencies to evaluate the correctness of new change. By taking into account the size of the product and the number of the involved developers, the average developer gets dozens of notifications during the week. This overloading by the notification causes their ignorance. In practice, the author of the commit usually asks for the review of his teammate by the chat. However, addressed teammates do not have to have the required competencies. This can cause a delay in the delivery of the commit by the protracted code review process. In a worse case, a teammate can approve the commit without verification of its correctness what can result in decreasing in source code quality and product error introduction.

Besides the implementation of the product functionality, this phase incorporates also the creation of the Unit tests and automated System tests. For the Unit tests source code developers use a special library. This library provides automatic execution and evaluation of the tests during the source code compilation. In the case of the System tests in addition to their implementation, it is also required to ensure their execution by their inclusion into the Continuous integration system described in the previous section 2.2. Also, System tests need to be recorded to the web application which is a unified tool for gathering and analysing tests within the whole product.

Another not less important part of the implementation phase is the writing of the technical documentation. For this purpose, project members are not used to creating JIRA tasks, but the creation of the documentation is an implicit part of the implementation process. Developer s are accountable for the content part of the documentation. A more detailed explanation of the documentation types, their form, content, and creation process is placed in the ending paragraph of the previous section 2.2.

2.3.5 Testing

Testing is the last step before the inclusion of new implementation into the shared repository and therefore into the module source code. Within the observed product, testing is performed on several levels. A more detailed description of the certain testing levels is summarized in the section that contains the introduction of the project 2.2. Only certain types of tests are used for verification of the implementation before its inclusion into the product. In concrete, Unit tests and Integration tests. After the commit is sent to the shared repository, the building of the modules is automatically triggered, which also includes the execution of the Unit tests. After successful verification of the change by the unit tests, next in the row are Integration tests. Their execution is also handled automatically. Besides this, there are scripts that validate a certain set of formatting rules and at the same time source code is analysed by the SonarQube tool (a tool for the static analysis of the source code [2]). In case that any of mentioned mechanisms evaluates commit as invalid, its author has to provides a solution according to the verification process result. Otherwise, if the commit is approved and reviewed by another team member, it can be embedded into a shared repository.

Besides the automated testing orchestrated by the Continues integration system, teams have in hand hardware resources as well as virtual machines, using which they can perform manual testing. Also, they can use those resources for the execution of the System tests. This kind of testing is usually more time-consuming, because of the necessity to configure hardware, or virtual devices. Though, this testing has a key value in order to verify the correct functionality of the module within the product. This higher-level testing is usually performed by the author of the implementation, Scrum Master, or another team member which has free capacity.

If the team needs to test the script for correction of the customer TR, testing is performed manually according to the solution design. Firstly, a configuration similar to the customer's is prepared on the testing environment, and afterward function of the script is verified. This special type of testing is always performed by someone who is not the author of the script.

2.3.6 Deployment

From the analysed process perspective, deployment stands for the inclusion of the commit, that holds implementation, in the shared repository. After delivery of the commit, the change request is considered terminated. If further testing discovers any error of the delivered implementation, correction of this issue is perceived as a new change request that enters the development process as a TR. Delivery of the implementation directly to the customers takes place in certain time intervals, which usually last one quarter or few months. These deliveries are named Releases. Release always includes multiple product changes, improvements, and new functionality. There are dedicated teams that are managing releases its delivery to the customers.

Delivery of the scripts for fixing of customer's TRs is performed in a special way. Along with the script, the team also delivers a detailly manual for script usage. Delivery is done using online sessions, with the attendance of a responsible team member, a technician from the customer's side, and the Customer's support member. A technician is following the manual and a responsible developer is there to help in case of any unexpected issue.

2.4 Roles and responsibilities of the process

Since the company is using a methodology based on the SCRUM, roles within the process have the same titles as SCRUM roles. Also, some of their responsibilities can overlap with the SCRUM definition. The team structure is illustrated in the picture below 2.3. There are four Development teams participating in the development process described in the previous section 2.3. Each team consists of 7-10 Developers including one Scrum master per each team, and one System Architect for the whole project. Also, there are two persons in the role of the Product Owners. They split the responsibilities of all four teams between each other's, so each is working and taking responsibility for his two teams. There are also different stakeholders which are serving input requests to the development process and at the end of the process, they are the customers of the process who are benefiting from the product increment.

2.4.1 Scrum Master

Each of the four teams from the observed project has his own Scrum Masters. People that represent this position were previously in the position of Developer. When previous Scrum Masters left their teams, management offered these positions to the Developers which had the biggest knowledge of the product. If addressed Developer

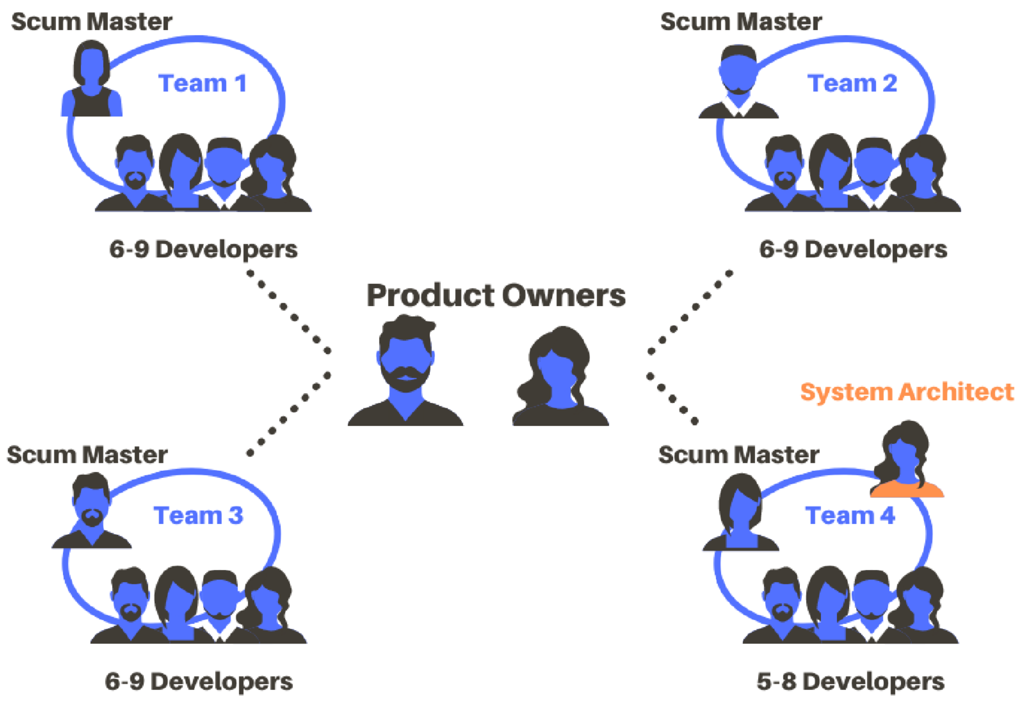

Figure 2.3: **The team structure of the project.** (Source: Own Creation)

agreed with the offer, he had to take an official course in order to get the certificate of the Scrum Master.

After the team member becomes a Scrum Master, he does not lose his previous responsibilities. He continues with all duties that Developers have. He has to work on the usual Developer's tasks which means he is analysing the incoming request, preparing solution designs, implementing the source code, test the new versions of the module, and other activities included within the development process. Besides that, as a person which the broadest knowledge of the product he is a person that makes decisions in terms of technical questions. For example, when there is the possibility to implement one feature using two different design patterns, which both have their advantages and disadvantages, Scrum Master decides which approach going to be used. Whit his knowledge he is also in charge of supporting Developers by providing help in the technical part of the task solving. And generally, he is sharing his product-related knowledge whenever the situation requires that. As an example can be used the arrival of a new team member. I this kind of situation Scrum Master is providing all technical information and explanations when it is needed.

On top of the responsibilities related to the technical part of the development, Scrum Master stands also in the role of a team leader and manager. He is managing the team and also coaching and motivating team members. He takes responsibility for the team tasks. He controls the fulfilment of the tasks and performance of the team members. He is administrating the team's backlog and divides tasks among the Developers. When the new TR arrives, he is one that chooses Developer to which will be the TR assigned. That means he must be aware of the team member's capacity. Reporting and communication of the team progress is another key responsibility that Scrum Master has. Also, he is organizing and moderating all events such as planning, daily meetings, and retrospectives.

2.4.2 Product Owner

Both Product Owners were previously in the role of the Scrum Masters. This implies that they have the deepest knowledge about the product out of the Development team and huge experience with the development process. Before they were promoted to the role of Product Owners, they had to pass an official course at the end of which they were provided with the certificate.

The Product Owner's responsibilities start at the beginning of the product development process where he is in charge of reception, analysis, and addressing of the incoming request. As it was described in the section documenting the development process 2.3, the person in this role has to provide a preliminary analysis of the task in order to find out if the task was addressed to the correct teams. If so, he assigns the newly received requests to the accountable team. If he comes to the conclusion that the task should belong to another area of the product, he addresses the task to the Product Owner of them that he believes is right. If he finds out that the task is based on the misunderstanding of the product functionality by the end-user, he informs the originator about the task rejection along with an explanation on how to solve his issue. If someone from the development team comes with his own requirement Product Owner can evaluate it and subsequently reject or approve it. If the requirement stands for significant change, he presents this request to the System council which is the authority with the right to approve such request. In the phase of the solution design, the Product Owner is appraising designs and, in situations, where there are more possible solutions, he has to make the decision which option will be preferred. Sometimes he is also creating the design solutions. As an experienced part of the team, he also prepares test scenarios from time to time. Occasionally, he is performing an analysis of the new functionality requests, which is originally the System Architect's job. When the less experienced Developers have some struggles or need advice with solving technical tasks they usually ask him for help.

A person in this role is the keeper of product-related knowledge. He gathers news about the product coming from all other areas outside the module and he is constantly spreading it around the teams. He is managing Product Backlog items as well as Sprint Backlog items. When it comes to planning, he always comes with a list of tasks that should be covered in the upcoming Sprint. This means he has to be aware of the team capacity, which he is reaching by communication with the team members and mostly with the Scrum Master. Also, he is monitoring team performance. He also works as a communication channel between stakeholders and the Development team. Therefore, he is responsible for fulfilling the Sprint Backlog and Product Backlog in front of the stakeholders.

2.4.3 Developers

There are four Development teams cooperating on the observed project. Each of the teams consists of 7-10 Developers. Most of the Developers are located in the same office, but each team has at least one member that works permanently remotely. One of the characteristics of the teams is that they are cross-functional. That means when we perceive a team as a single unit, its knowledge covers a variety of areas and it is able to cover all tasks that the software development process includes. Each of the team members has at least basic knowledge of all particular development process operations, and according to his specialization and experience, he is profiled on a specific part of the process.

Developers represent the heart of the whole development because they are responsible for fulfilment of the Product Backlog tasks and therefore, they are real creators of the product. Their daily business incorporates working on the tasks from the Sprint Backlog. They analyse requests, conduct solution designs, write the source code, formulate documentation, execute manual testing, and control or extend automated testing. Besides these crucial activities, they are communicating with the task originators, track their progress within the JIRA system, and help each other with the Developer's everyday matters. When they see a space for improvement in the product as well as in the working process, they can share the ideas with the team and responsible authorities.

2.4.4 System Architect

Currently, there is one System Architect representing the whole project, or in other wording, one Architect is covering the area for the entire module. He was previously working as a Developer and he was promoted to the position of Architect according to his competencies and performance when the position was opened. His main assignment is to provide a detailed analysis of the requirements for the new features and the creation of the corresponding documents. More information about the document and its purpose can be found in the section describing the development process 2.3.

As he is a member of the development team, in case he has the free capacity, he is performing the same tasks as Developers do. Also, he is in closer contact with all Architects from other modules and with the System council. They are discussing all new changes with each other, share ideas, evaluate their analysis, present new features, etc. A person in this position is therefore in charge of providing the information from other modules to the Development teams.

2.4.5 Responsibilities within the process

All four roles previously introduced in this section have different responsibilities towards the tasks of the development process. Since the process of the product development described in the section above 2.3 is complex and incorporates a higher number of tasks, this section contains RACI (Responsibility assignment) matrix 2.1 that summarizes all the tasks. Structure of the matrix consist of the task description placed in the first column. This column is followed by the other four columns, which stand for the four roles of process. Each row then represents one specific task and the cell belonging to the column of the role defines the type of responsibility that role has to the task. Particular responsibilities are stated by four letters 'R', 'A', 'C', and 'I'. Acronym 'R' marks the role or the person who executes the task. Letter 'A' in the column means the role is approving work done by 'R'. The 'C' is used for roles that are communicating with the 'R' during the task execution and their opinions are sought. The last letter 'I' belongs to the roles that are only informed about the task completion.

2.5 Events and artifact of the process

As the company is currently using for the project a methodology based on the SCRUM, also time structure of the process, its events, and existing artifacts are similar. But as it is explained in this section, in certain cases similarity can be only in the name of the event and all other aspects can differ from the SCRUM definition. The main event is the Sprint, which more or less follows the definition of the SCRUM Sprint. There are four other events that are periodically repeated during each Sprint. In concrete, Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective. All events are described in separate paragraphs. Each characterization contains a list of participants, their roles within the event, and activities executed by the time of the event. In the second half of the section, there are paragraphs depicting two artifacts of the development process Product Backlog and Sprint Backlog. This content provides information about the form and purpose of the artifacts as well as a summary of the ways how certain roles can manipulate the artifacts.

2.5.1 Sprint

Every Sprint lasts three weeks. At the beginning of the sprint, usually on the first or the second day of the Sprint, Sprint Planning is happening. During this meeting, the team discusses their tasks for the upcoming three weeks. Developers then work on specified tasks. Each day of the Sprint all four teams have their private Daily Scrum. And at the end of the Sprint, there is a meeting called Sprint Review, during which all four development teams, their Scrum Masters, and Product Owners gather, and each team presents work that they did during the last three weeks. Also, at the end of the Sprint teams can have their private Sprint Retrospective.

2.5.2 Sprint Planning

This event lasts around 45 minutes with the attendance of the Developers, Scrum Master, and Product Owner. Participants are using a shared text document that

Table 2.1: **RACI matrix summarizing roles of the development process and their responsibilities according to the tasks within the process.** (Source: Own Creation)

Task	Developer	Scrum Master	Product Owner	System Architect
Assignment of the task to the team	I	C	R, A	I
Preliminary verification of the TR area	I	I	R, A	I
Readdressing of TR	I	I	R, A	I
Verification if the TR is real product issue	C	C	R, A	C
Explanation of the functionality to the TR originator	R, C	R, C	R, A, C	R, C
Assignment of the task to the team member	R, C	R, A, C	R, C	R, C
Detailly analysis of the TR, and its artifacts	R, C	R, C	R, A	R, C
Monitoring of the System testing results	R	R, A	R, A	R
Creation of the system testing related TRs	R	R, C	A, C	R
Resolving of the Environment issues	R	R, A, C	I	R
Creation of the refactoring or redesign request	R, A	R, A	C	R, A
Approval of the refactoring or redesign request	C	C	R, A	C
Detailly analysis of the refactoring or redesign request	R	R	A, C	R
Creation of the solution design content	R, C	R, C	A, C	R, C
Creation of the solution design JIRA representation	R, C	R, C	A	R, C
Creation of the test design	R, C	R, C	A, C	R, C
Approval of the solution design	C	C	R, A	C
Description of issue in the Limitations and Workarounds document	R, C	R, C	R, A, C	R, C
Creation of the testing and deployment scenario for the customer's TR	R, C	R, C	R, A, C	R, C
Implementation of the change into the source code	R, C	R, C	A, C	R, C
Providing of the code review	R, A, C	R, A, C	I	R, A, C
Implementation of the tests	R, C	R, C	A, C	R, C
Inclusion of the tests to the Continuous Integration system	R, C	R, C	R, A, C	R, C
Recording of the implemented tests into system	R, C	R, C	R, A, C	R, C
Creation of the technical documentation	R, C	R, C	R, A, C	R, C
Execution of the manual testing	R, C	R, C	A, C	R, C
Delivery of the scripts to the customers	R, C	R, C	A, C	R, C

contains a list of tasks that need to be done for certain Sprint. This list is created by the Product Owner and each item that is representing the task contains a short description, level of priority, and link to the JIRA task item. At the beginning of the

Planning, the team runs through the list of tasks from the previous Sprint. Scrum Master is here in the role of the moderator, which comments on each task or asks Developers who are assigned to the task to comment current task status. Simultaneously, Scrum Master is making notes about the task status and then continues with the next one. After all tasks were reviewed, Scrum Master asks the Product Owner to present a new list of the task for the current Sprint. The Product Owner explains each task and his priority. All other participants are free to ask about needed details or discuss all aspects of the tasks. Then Product Owner takes unfinished tasks from the previous lists and incorporates them into a list for the current Sprint. Then, he can also update the priorities of the tasks. When the team has a clear perception of the tasks and there are no more questions to the Product Owner, Product Owner he can leave the meeting. Afterward, Scrum Master is walking through the tasks in the new list and Developers can ask for the assignment of the task, if they are interested. If nobody wants to take the task, Scrum Masters try to find someone by discussing with the Developers. At the end of this procedure, each task from the list has at least one assignee, which means that the Sprint Planning is done. It is important to mention that the Development team is not discussing the size of the workload and always receives all tasks that the Product Owner listed in the document.

2.5.3 Daily Scrum

According to the calendar event representing the Daily Scrum, it should last 15 minutes, and it occurs every workday. Every team organizes its own Daily Scrum separately. Participants of this meeting are Developers, Scrum Master, and the Product Owner. This short gathering should be the time when each Developer informs other participants about the status of the task he is currently working on. What he did since the last Daily Scrum and what he will do until the next one. In practice, depending on the personality of the particular Developer and his current task, developers tend to provide very deep explanations and use this space to discuss possible solutions with the Product Owner or Scrum Master. Finding a solution for more difficult problems can end up in a long discussion between two or three participants and all other participants are more or less wasting their time. This kind of discussion very often causes an exceeding of the 15 minutes time frame.

2.5.4 Sprint Review

Sprint Review is the only event where all four development teams their Scrum Masters and Product Owners meet at the same time. The agenda of this meeting is that each team is presenting the work, they did during the last Sprint. It always takes place on the last day of the Sprint. Either the team's Scrum Master, chosen Developer, or a few of the Developers are presenting work finished in the last three weeks. Depending on the presenter and last Sprint's tasks, the presenter is showing the product increment which can have a form of the new feature and optimization or change of the previous functionality. In the case of new features, it is usual to have a demo presentation that includes an explanation of the new functionality and a live demonstration of feature usage on testing hardware. Since the significant part of the workload of the teams is bug fixing, demo presentation is not done by each team. Therefore, teams are used to present what issues they fixed during the Sprint. Taking into account the fact that there is no moderator in this event, it is very often that presenters are providing very long and extremely detailed descriptions of every issue they corrected. This way is very time-consuming and does not have a beneficial effect on the meeting participants. Sometimes this meeting is more about showing off with the delivered work, than presenting useful facts to the other project colleagues.

2.5.5 Sprint Retrospective

This is a private meeting where only Scrum Master and Developers are invited. Scrum Master asks each of the Developers about his feelings due to the last Sprint. Developers have space to evaluate all aspects such as Sprint Backlog tasks and their characteristics, working process, working environment, tools used during the development, team's mood, their personal progress, the performance of the team or particular team members, and so on. This meeting is lead in form of a discussion and everyone can come out with his opinion on the currently discussed topic. Scrum Master is also sharing his feeling and opinions as he also works as a Developer. Through the discussion, the team is tiring to find solutions and answers for problems that pop up during the meeting. Scrum Master is making notes about all important topics and if it is needed, he is trying to the solve appeared issues with the Product Owner or company management. It is not very usual that team members have significant issues. And if they have some, the issues are mostly principled or systemic problems and therefore cannot be removed easily.

2.5.6 Product Backlog

Product Backlog is a list of all tasks from the JIRA system that are labelled as tasks of the observed module. Since tasks are materialized as JIRA system records, they are visible and accessible to each member of the project. Also, each project member has access right to create a new task within the JIRA system. This implies that each team member can add a new task into the Product Backlog. Also, employees that are working on the other modules are free to create new Product Backlog items. Customer Support technicians are another group of people that has this possibility. This fact causes that Product Backlog often contains duplicate, wrongly formulated, inaccurately addressed, incompletely filled, and unnecessary tasks.

2.5.7 Sprint Backlog

As it was already mentioned in the section about the event named Sprint Planning, Sprint Backlog has two representations. The first is a document formulated by the Product Owner, and the second is a set of JIRA task records that are reflecting the document. JIRA items are labeled with the Sprint identificatory in order to provide an association with the particular Sprint. This collection of tasks contains unresolved ones from the previous Sprints as well as tasks that were newly added by the Product Owner. Each task has its priority, description, status indicator, and possibly its assignee. Similarly, as it is in the case of the Product Backlog, everyone with access rights to the JIRA system is free to add a new task to the Sprint Backlog. This of course means that if a concerned person does not update a shared document, these two representations can differ. This sometimes causes that there are tasks that team members are not aware of, or Developers are working on tasks that they shouldn't work on according to the prioritizing.

2.6 Analysis summary

The process of product development, described in the section 2.3, consists of several phases. Each phase incorporates several different tasks. Some of the tasks are executed only by specific roles and some of them can be performed by all the project members. Fact that the company is currently using a methodology based on the SCRUM results in the situation where the roles that appear in the process, ongoing events, and existing artifacts of the process may look like they follow the definition of SCRUM. But, from the closer perspective that is provided within this chapter 2, it is apparent that the current methodology differs from the SCRUM definition. In some cases, the difference is significant and in other cases, we can find only minor contrast. From a slightly more concrete point of view, when we take into account the responsibilities of all project members described in the section 2.4 of this chapter, we can see roles have some accountabilities in addition. Artifacts and events used within the process also require some corrections in order to fulfil the SCRUM definition. Allocated time frames and placement within the Sprint are similar to the definition, but their agenda and the activities of some participants are different. Artifacts are missing some basic characteristics, and the possibilities of how some roles can manipulate them are also incorrect comparing to the definition.

Scrum Master

In this process person in the role of the Scrum Masters in practice stands for several roles. The first role is the role of the Developer, which means a person in this position is executing all tasks that provide increment on the product. The second is the role of the team leader which means supervising and controlling other Developers. And at the top of that, an employee in this role is also fulfilling some duties listed in the SCRUM definition.

First of all, the person in this role should not act as a Developer within the process. Someone could say that there is no explicit part of the SCRUM definition that is prohibiting the Scrum Master's participation in the development activities. But, since the main goal of this role is an establishment of the SCRUM methodology on the project and among its members, which is currently not completely achieved on the project, it is crystal clear that this goal requires more effort what implies this role cannot be combined with the role of the Developer.

The Scrum Master's role of the team leader has two parts. The first part is coaching and motivating the team members. This is in line with the definition of the SCRUM. The second part is his authority towards the technical sphere of development. As it is mentioned in this subsection, Scrum Master should put all his effort into the responsibilities listed in the definition. Thus, technical tasks and decisions must be done by other participants of the process. Also, the responsibility of the task fulfilment needs to be removed from this role. Developers with the widest knowledge base should be rather promoted to the Technical Lead roles or other roles except for Scrum Master. Administration of the team's backlog is completely out of the Scrum Masters' scope of activities. The dividing of tasks among Developers is also an unwanted activity that is decreasing the team's self-organization, what is the opposite effect that should this role bring into the team's environment. Organization of all events is correctly assigned to this role, but as it is more detailly described in the further subsections, Scrum Master needs to put more focus into the agenda of meetings, as well as activities executed by particular team members during the events that differ from the definition.

Product Owner

One of the Product Owner's functions is the management of the Product Backlog. Thus, preliminary analysis of the incoming requests is correctly addressed to this role. This statement can be defended by an explanation that by accepting the task, the Product Owner is adding a new item into the Product Backlog. This rule can be also applied to situations when the Product Owner evaluates change requests from the Developers. In case a request may have a significant effect, the Product Owner forwards this proposal to the System council. This can be perceived as an obtaining of the requirements from the stakeholders and transforming them to the Product. Approval of the solution designs can stay in the competencies of this role because it can be understood as clarification of the Product Backlog items. Creation of the solution designs, preparation of the test scenarios, and analysis of the new functionality requests are activities that should be done by someone else than the Product Owner. Helping the Developers with their technical struggles should be rather done by more experienced members of the Development team. Maintenance and sharing of the product-related knowledge are also acceptable for this role as well as management of the Product Backlog and Sprint Backlog.

With a closer look into the RACI matrix 2.1 that accumulates all tasks of the development process, we can see that person in the role of Product Owner is executing several tasks that he should not. In concrete he or she should not be responsible for these tasks:

- Assignment of the task to the team member
- Detailly analysis of the TR, and its artifacts
- Monitoring of the System testing results
- Creation of the system testing related TRs

- Description of issue in the Limitations and Workarounds document
- Creation of the testing and deployment scenario for the customer's TR
- Inclusion of the tests to the Continuous Integration system
- Recording of the implemented tests into system
- Creation of the technical documentation

Developers

As it is written in the section describing the Scrum Developers 1.3.5, two key characteristics of a functional Development team are cross-functionality and self-organization. The content of previous sections of this chapter confirms the presence of cross-functionality between the developers of all four teams. But when it comes to the self-organization of the teams there is space for improvement. The fact that Developers are able to choose which task they want to work on can be perceived as a trait of self-organization. But on the other hand, the situations when tasks are assigned by the Scrum Master or Product Owner predominate. Thus, abandoning this habit would be a step further towards the alignment with the SCRUM definition.

All tasks that Developers execute during the Sprint, and produce Increment on the product are correct. When it comes to Sprint Planning, Developers are provided with the list of tasks gathered by the Product Owner, and they have to accept this list as a Sprint Backlog. SCRUM definition says that Sprint Backlog must be created by the Developers, not by the Product Owner. So, this is another point on the list of changes that need to be done in order to apply the SCRUM methodology.

System Architect

This role is completely custom, and there is no such definition of the role of System Architect within the SCRUM. But the rule that says the Development teams are cross-functional, and domains of the particular team members incorporate all needed specializations approves that the person in the role of System Architect can be perceived as a Developer. This implies that all responsibilities of the Developers can be also responsibilities of the System Architect. In the end, there is flexibility in terms of what custom roles can exist within the process that uses SCRUM methodology, without breaking fundamental principles of the SCRUM.

Sprint

Duration of the Sprints and all events that are ongoing during each Sprint fit the definition formulated within the SCRUM. Some corrections can be done in the scheduling of particular events that are happening within each Sprint. The Sprint Planning should be the first event that opens new Sprint and therefore should not take place two or three days after the Sprint beginning. Also, as it outcomes from the provided analysis, Sprint Retrospective is not a mandatory event and thus can be skipped by the team's or Scrum Master's decision. Making Sprint Retrospective a mandatory event of each Sprint could bring improvement into the process of product development.

Sprint Planning

Starting with the evaluation of the time frame reserved for this event, in the current situation it is enough wide for the execution of all planning activities. But with the erasing of some distinctions from the SCRUM definition, this time frame may need to be extended. The list of participants is in accordance with the definition.

The first question that should Sprint Planning answer is 'Why is this Sprint valuable?', is more or less fulfilled in the current form of Sprint Planning. Since the Product Owner brings the list of the new tasks, that need to be done during the Sprint, as the author of the list he or she can present why are these tasks valuable.

The second question 'What can be Done this Sprint?', which should be resolved by Developers through the discussion with the Product Owner, is currently not resolved correctly. The contemporary way of defining the Sprint Backlog is done without the right of the Developers to discuss a number of tasks or the presence of the task in the list. The Product Owner is just providing the list of tasks that must be accepted. Therefore, there is no need for estimation of the work that should be done. This autocratic approach in planning removes principles of empiricism and therefore should be removed.

The third question that should be answered during this event is 'How will the chosen work get done?'. Since analysis contains information, that Developers are able to discuss tasks from the provided list with the Product owner, this action is partly done. But there is no exact formulation of the Definition of Done for particular tasks, what can result in situations that Developers need additional explanation during the development. Or in a worse situation, this can cause delivery of incorrect functionality due to misunderstanding and lack of Definition of Done. So, it is more than required to add compulsorily of formulation of Definition of Done for each task within the Sprint backlog.

Daily Scrum

The agenda of this meeting and the list of participants is correct compared to the SCRUM definition. As a bottleneck of this event can be marked fact that participants sometimes start a deep discussion about a particular topic and therefore meeting exceeds a timeframe of 15 minutes. This needs to be eliminated by the Scrum Master which should interrupt this kind of dialogue and remind discussants that they can continue outside of the Daily Scrum meeting.

Sprint Review

As it comes from the SCRUM definition, this meeting should serve as space where teams present their work to important stakeholders and modifications of plan and Product Backlog are done. The analysis discovered that the content of this meeting covers only one part which is the part where all teams present their output of the work. Also, it was mentioned that there is no moderator who is controlling the agenda of the event. This can be done by one, or more Scrum Masters. There is not any presence of plan or Backlog adjustments within the current state. This is done at the beginning of each Planning. Taking into account fact that this meeting is attended by all teams, it would not make sense to discuss such modifications with each team one by one. So, a possible solution can be a separation of the presentation part into another meeting which would be common for all four teams. Then Backlog and plan updates would be done separately by each team.

Sprint Retrospective

The list of participants which in the current situation incorporates Developers and Scrum Master can be extended by the invitation of the Product Owner. The agenda and the basic idea of this meeting are aligned with the definition of SCRUM and therefore there are no significant characteristics that need to be changed. Maybe more effort from the site of company management could be added into the process of solving issues that were discovered during this meeting.

Product Backlog

Product backlog that is represented by records within the JIRA system is reaching enough level of visibility and transparency that it should have according to the definition. Also, the possibility of the team members to create and update JIRA records follow the rule of refinement of the tasks. But maybe there should be applied some policies how to create a task which should bring more clarity and organization into this artifact.

Sprint Backlog

As it is written in the above section discussing Sprint Panning 2.6, this artifact is built exclusively by the Product Owner. This is an incorrect way and formulation of the Sprint Backlog needs to be done by Developers and their discussion with the Product Owner. Also, a form of the Sprint Backlog or better says two separate sources are not the best option for how to materialize the Sprint Goal. A possible solution is to use only JIRA records that are accessible for all project members and which enables the creator to capture important features of each task such as estimated duration, priority, description, and so on. Also, the rights of the project members to add tasks into Sprint Backlog should be limited. This could be solved maybe by adding the requirement of approval before the task will be included in the Sprint Backlog. That would mean only team members would be able to decide if the new task can be included in the current Sprint Backlog.

Chapter 3

Proposal of solution

This chapter accommodates a proposal of solution that needs to be applied to the current state of the observed project to achieve implementation of the SCRUM methodology. The proposal is based on the analysis of the contemporary situation which is provided in the previous chapter 2. Changes that are summarized within this proposal were formulated using theoretical information from the chapter that contains a theoretical review of the problem 1. More concrete, the first section 3.1 is describing changes that need to be done in the team structure, all project roles and their responsibilities, Sprint structure, and its events, and artifacts. The second section 3.2 evaluates the financial side of the proposal. Within this section, there is the summarization of available educational courses ad their prices as well as costs of the project members' labour. Then there is provide calculation of the costs based on previously gathered information, which involves costs of the project members' education and costs of salaries of the new project members that are introduced in the proposal. The last section of this chapter 3.3 outlines the benefits of the proposed solution.

3.1 Proposal of changes

This section contains the proposal of changes that need to be implemented to the processes within the observed project in order to follow the definition of the SCRUM. All proposals are made with the aim of optimization of the working process and retention of mechanisms of the development process 2.3 that needs to stay unchanged because of characteristics of the product 2.2 and the company's structure 2.1. The proposed solution is based on the information gathered in the Analysis of the contemporary situation 2, with the intention to erase shortcomings highlighted in the Analysis summary 2.6. The first half of this section contains the proposal of changes toward the team structure and its roles. The second part is dedicated to the corrections of events and artifacts of the process.

3.1.1 Team structure

According to the analysis, which is described within the previous chapter 2, there are four different roles within the process of product development. In the current structure 2.3, there are two Product Owners, and four teams that consist of 7-10 Developers, each team has its own Scrum Master and there is one System architect, that also acts as a Developer. The newly proposed structure displayed on the picture 3.1 should consist of two Product Owners, two Scrum Masters, and four Development teams. Each team will consist of 6-9 Developers and one Team Leader and there will be one System Architect as it is in its current state.

The newly introduced role of the Team Leader will be a role that incorporates responsibilities of the Developer and accountabilities that were identified as unwanted for the role of Scrum Master within the section of Analysis summary 2.6. The number of Developers was decremented by one person because the Team Leader will also act as Developer and therefore keeping the Development team's size results in this situation.

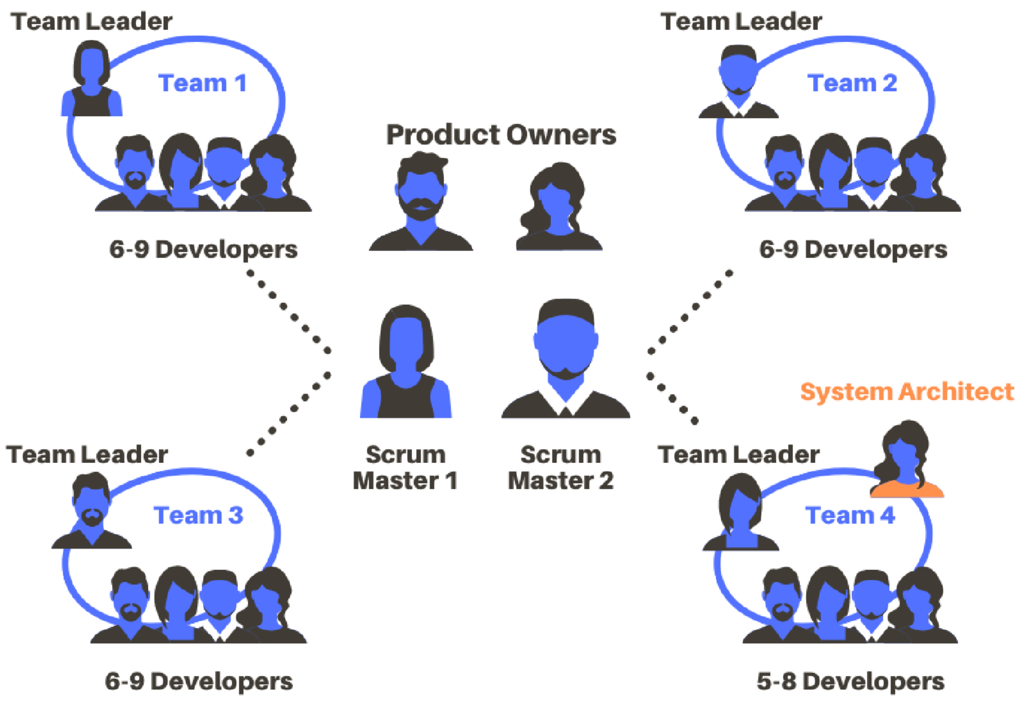

Figure 3.1: **Proposed team structure of the project.** (Source: Own Creation)

The number of the Scrum Masters will be decreased from four (one per team) to two (one per two teams). This decision was done with considering of fact that Scrum Master's work will be correct implementation and control of compliance of the SCRUM principles, without any participation in technical tasks of the development. This will lower the number of Scrum Master's tasks, and therefore one Scrum Master can serve two teams at the same time. Also, the financial perspective affected this decision, because there are two more employees in the new team structure, which means extra costs for the company.

3.1.2 Scrum Master

As it is mentioned in the previous subsection 3.1.1, a person within the role of the Scrum Master will be deprived of his development activities and his authority towards the technical decisions that he is serving in the current state. By removing the Developer's duties, Scrum Master will be able to completely focus on the implementation of the SCRUM methodology and its principles to the development process. An employee in this role is in charge of SCRUM rules application and therefore he or she must put all his or her effort into controlling SCRUM policies fulfilment. In terms of technical leadership and decision making that is Scrum Master providing in the current state, all these operations will be readdressed to the newly introduced role of the Team Leader.

Coaching and motivating of the team members are correctly assigned to this role and should be kept also in the proposed solution. Accordingly, the organization of all SCRUM events should stay on the list of Scrum Masters responsibilities. During these events, he will be in the role of the moderator, which will control if the meeting agenda is correct, if the list of participants fits the event definition, if the time frame and location within the Sprint is accurate, and if all participants represent correct roles within the event. And the last but not least, he or she will be observing how is manipulated with the Scrum Artifacts, and in the case of any differences from the SCRUM definition he will take an action for the correctness of the situation.

According to the particular tasks that exist in the process of the product development, and Scrum Masters' responsibilities towards them, there is no task that would person in this role execute or which execution would he or she approve. All tasks and Scrum Master's relation to the is depicted in RACI matrix 3.1.

3.1.3 Product Owner

The main accountabilities of the employee in the role of the Product Owner are maximization of the product value, management of the Product Backlog, and definition and promotion of the Product Goal. Also, he or she has responsibilities towards the visibility and accessibility, transparency, and intelligibility of the Product Backlog. According to the summary of the analysis which focuses on the role of the Product Owner 2.6, Product owners are correctly achieving these objectives in the current state of the project. Also, as follows from mentioned part of the Analysis summary, there are several tasks that will not be executed by the Product Owners in the proposed solution. One thing that needs to be granted to the Product Owner is a better way how to control Product Backlog. This means that the Product Owner will be an authority that will approve recently created tasks as Product Backlog tasks. The mechanism for implementation of this rule will be detailly described in the subsection about the proposals for the Product Backlog 3.1.13.

In concrete, tasks that the process of the product development consists of, and the Product Owner's responsibilities against them are listed in the RACI matrix 3.1. In the proposed structure will be an employee in this role in charge of the execution of these tasks:

- Assignment of the task to the team
- Preliminary verification of the TR area
- Readdressing of TR
- Verification if the TR is real product issue
- Explanation of the functionality to the TR originator
- Approval of the refactoring or redesign request
- Approval of the solution design

Besides the responsibilities of an executor, the person in this role stands also for an authority that is evaluating and approving the correct execution of the tasks, which was provided by other team members. This list summarizes particular tasks, where the role of the Product Owner acts as an approver:

- Assignment of the task to the team
- Preliminary verification of the TR area
- Readdressing of TR
- Verification if the TR is real product issue
- Explanation of the functionality to the TR originator
- Creation of the system testing related TRs
- Creation of the refactoring or redesign request
- Approval of the refactoring or redesign request
- Detailly analysis of the refactoring or redesign request
- Creation of the solution design content
- Creation of the solution design JIRA representation
- Creation of the test design
- Approval of the solution design
- Description of issue in the Limitations and Workarounds document
- Creation of the testing and deployment scenario for the customer's TR
- Implementation of the change into the source code
- Implementation of the tests

- Inclusion of the tests to the Continuous Integration system
- Recording of the implemented tests into system
- Creation of the technical documentation
- Execution of the manual testing
- Delivery of the scripts to the customers

3.1.4 Developers

By closer look at the two key attributes that each Development team should have and with the consideration of facts written in the analysis summary focusing on the Developers 2.6, we can state the presence of cross-functionality within the current state of the project is sufficient. When it comes to self-organization among the team members, this peculiarity will be improved by the removal of the ability of task assignments from the list of Scrum Master's and Product Owner's competencies. Team members will divide the tasks between each other by themselves with the help of the Scrum Master and Team Leader. It is necessary to mention, that Scrum Master will only take care of the procedural side of the assignment. The Team Leader as a part of the Development Team will be an authority, which will be able to decide who will execute a task, but only in borderline cases. Most of the time Developers will be free to take tasks that they will consider as the right candidate according to the priorities, state of the Sprint Backlog, and Sprint Goal.

Another change that needs to be applied comparing to the current state is that Developers need to be creators of the Sprint Backlog. This means they will actively participate the Sprint Planning where they will construct a list of the tasks that will materialize the Sprint Backlog. Their participation during the process of planning will be again monitored by the Scrum Master.

In order to provide a complete list of tasks that are Developers responsible for during the process of product development, we can take a look into the RACI matrix 3.1. As it comes from SCRUM definitions, Developers are those team members who execute tasks from the Backlog, which results in the fact that they are marked as executors almost in all tasks of the development process. They are no executors only in case of tasks that are part of the Product Backlog management. In particular, they should execute these tasks in the proposed solution:

- Explanation of the functionality to the TR originator
- Assignment of the task to the team member
- Detailly analysis of the TR, and its artifacts
- Monitoring of the System testing results
- Creation of the system testing related TRs

- Resolving of the Environment issues
- Creation of the refactoring or redesign request
- Detailly analysis of the refactoring or redesign request
- Creation of the solution design content
- Creation of the solution design JIRA representation
- Creation of the test design
- Description of issue in the Limitations and Workarounds document
- Creation of the testing and deployment scenario for the customer's TR
- Implementation of the change into the source code
- Providing of the code review
- Implementation of the tests
- Inclusion of the tests to the Continuous Integration system
- Recording of the implemented tests into system
- Creation of the technical documentation
- Execution of the manual testing
- Delivery of the scripts to the customers

3.1.5 Team Leader

This is a completely new role that does not appear in the current state of the project. Persons that should stand for this role are employees that are in roles of the Scrum Masters within the current project team scheme. They are the right employees to occupy this position since they have a high level of technical skills and wide knowledge about the product as well as experience with a leading position. If other, less experienced Developers will face some obstacles during the execution of the tasks Team Leader is one that will help them. Also, in situations where there is a need to decide between two or more technical solutions, Team Leader will be one that holds the final decisions on his shoulders. His or her assignments will not be related only to decision making, but he or she will be also a full-featured Developer 3.1.4 which will execute all actions that result in the product increment.

In the more concrete wording, Team Leaders will execute all tasks that Developers 3.1.4 executes, which is visualized within the RACI matrix 3.1. Moreover, employees in this role will be in charge of approvement of the fulfilment of these particular tasks of the development process:

- Assignment of the task to the team member
- Detailly analysis of the TR, and its artifacts
- Monitoring of the System testing results
- Resolving of the Environment issues
- Providing of the code review

3.1.6 System Architect

By the fact that this role is an extra-defined role that is not part of the SCRUM definition, we can consider the role of the System Architect as a special domain of competence of the Developer 3.1.4. Thus, there is no need to define special rules what should or should not person in this role execute within his or her existence in the product development process. All rules that belong to the role of Developer also apply to the role of System Architect.

3.1.7 Responsibilities within the process

Except for the proposal of general activities and accountabilities that each role has to fulfil within the proposed changes, there is also needed adjustment of responsibilities towards the particular tasks of the development process. For these purposes, this subsection contains RACI matrix 3.1, which incorporates the same list of tasks as a RACI matrix 2.1 locate within the subsection of the previous chapter, which is dedicated to the analysis of the current assignment of responsibilities within the observed process 2.4.5. Comparing to the mentioned matrix 2.1, there are several changes to the table structure. In particular, there is a new column dedicated to the role of Team Leader 3.1.5. Column with title 'Developer' stands for responsibilities of the Developers 3.1.4 as well as System Architect 3.1.6. These two roles were considered unifiable roles from the SCRUM definition point of view. The other three columns of the structure in a concrete column with the tasks of the process and column for the Scrum Master's and Product Owner's responsibilities keep the same semantic as in the previous RACI matrix 2.1.

If one takes a closer look at the content of the RACI matrix situated below 3.1, it is apparent that there are several differences in the assignment of particular relations of the roles towards the tasks. The most significant changes are proposed for the role of the Scrum Master, which was completely deprived of execution of tasks which are marked by acronym 'R'. Also, the column capturing the relation of this role according to tasks does not contain the letter 'A'. Removal of both these types of responsibilities implies that the person in the role of Scrum Masters acts as an advocate of SCRUM methodology and not as a technical worker.

Table 3.1: **RACI matrix containing proposal of roles of the development process and their responsibilities according to the tasks within the process.** (Source: Own Creation)

Task	Developer	Scrum Master	Product Owner	Team Leader
Assignment of the task to the team	I	C	R, A	C
Preliminary verification of the TR area	I	I	R, A	C
Readdressing of TR	I	C	R, A	C
Verification if the TR is real product issue	C	I	R, A	C
Explanation of the functionality to the TR originator	R, C	I	R, A, C	R, C
Assignment of the task to the team member	R, C	C	I	R, A, C
Detailly analysis of the TR, and its artifacts	R, C	I	I	R, A, C
Monitoring of the System testing results	R, C	I	I	R, A, C
Creation of the system testing related TRs	R, C	C	A, C	R, C
Resolving of the Environment issues	R, C	I	I	R, A, C
Creation of the refactoring or redesign request	R, C	I	A, C	R, C
Approval of the refactoring or redesign request	C	I	R, A	C
Detailly analysis of the refactoring or redesign request	R, C	I	A, C	R, C
Creation of the solution design content	R, C	I	A, C	R, C
Creation of the solution design JIRA representation	R, C	C	A, C	R, C
Creation of the test design	R, C	I	A, C	R, C
Approval of the solution design	C	I	R, A	C
Description of issue in the Limitations and Workarounds document	R, C	I	A, C	R, C
Creation of the testing and deployment scenario for the customer's TR	R, C	I	A, C	R, C
Implementation of the change into the source code	R, C	I	A, C	R, C
Providing of the code review	R, C	I	C	R, A, C
Implementation of the tests	R, C	I	A, C	R, C
Inclusion of the tests to the Continuous Integration system	R, C	I	A, C	R, C
Recording of the implemented tests into system	R, C	I	A, C	R, C
Creation of the technical documentation	R, C	I	A, C	R, C
Execution of the manual testing	R, C	I	A, C	R, C
Delivery of the scripts to the customers	R, C	I	A, C	R, C

3.1.8 Sprint

The timeframe allocated for the Sprint, which lasts three weeks, works very well with the types and time requirements of tasks that exist within the development process and therefore does not require any adjustments. When it comes to the time location of the events of the Sprint there are some changes that need to be introduced. As it is depicted on the visualization of the Sprint and its events 3.2, the very first event of the Sprint will be the Sprint Planning. This event will begin new Sprint and therefore will take place on the first day of the Sprint. Then there will be Daily Scrum which will happen every day at the time agreed by the team members. Sprint Review as a penultimate event of the Sprint, along with the Sprint Retrospective which is the last event of the Sprint will be organized on the last day of the Sprint. All four mentioned events, Sprint Planning, Daily Scrum, Sprint Retrospective, and Sprint Review will be obligatory and could be skipped on in exceptional situations. The person that will guarantee their realization and their correct location within the Sprint will be Scrum Master.

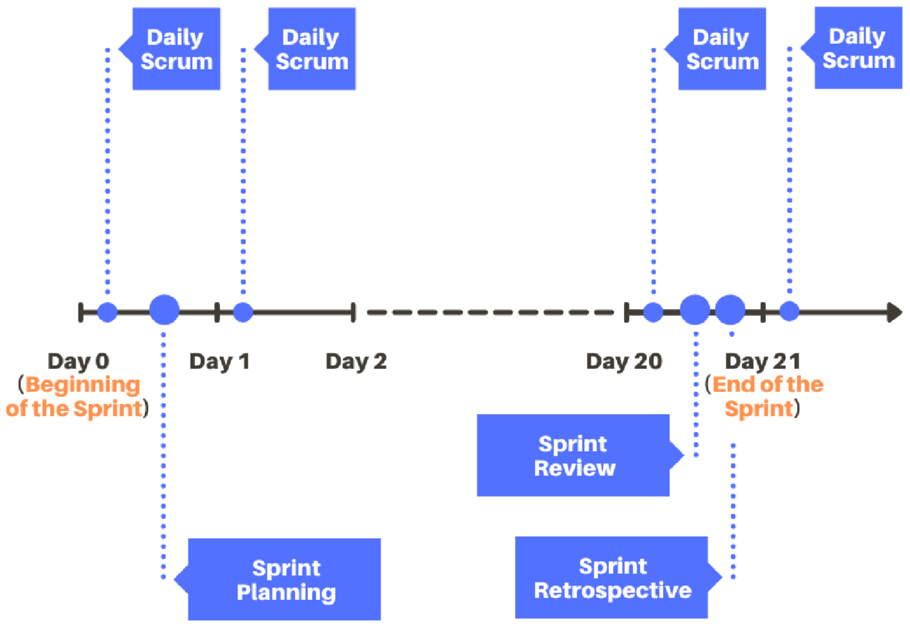

Figure 3.2: **Sprint and its events.** (Source: Own Creation)

3.1.9 Sprint Planning

Starting with the duration of this event and list of participants, which both do not have to be changed comparing to the current state, we can say that this event will last 45 minutes (or even more when the size of the agenda will require it), and the attenders must be Product Owner, Scrum Master and all Developers of the team. Important change needs to be implemented in the time location of this event within Sprint. As it is explained within the subsection about the Sprint 3.1.8, and visualized in figure 3.2, Sprint Planning needs to be placed at the beginning of the Sprint.

By closer look at the content of the meeting, Sprint Planning needs to achieve resolution of three topics - ‘Why is this Sprint valuable?’, ‘What can be Done this Sprint?’, and ‘How will the chosen work get done?’. As it was concluded in the part of the Analysis summary focused on Sprint Planning 2.6, the answer for the first topic is correctly provided in the current state and therefore does not require any adjustments.

The second question ‘What can be Done this Sprint?’ will be resolved by the Developers in the proposed solution. Because of the character of the development process and the way how are incoming change requests processed, there cannot be done any significant change in the fact that the Product Owner brings the list of the task that needs to be done. But what needs to be changed is the way how these tasks from the list are transformed to the Sprint Backlog items, as well as what number of tasks will form the list be accepted. Focusing on the crucial characteristic of the SCRUM, which is empiricism, only Developers will estimate the time constraint of the tasks. This will be achieved by the discussion between Developers and Product Owner, which will result in a clear definition of what needs to be done within the task and how it can be done. After this analysis of the whole list of the tasks, Developers together with the Product Owner will negotiate what is possible to deliver within the upcoming Sprint. Product Owner will highlight priorities of the particular tasks and with their consideration, Developers will establish Sprint Backlog.

The third topic which incorporates specification of how selected tasks will be implemented, and what will be considered as required work will be also resolved with the application of proposals written above in this subsection. There will be a focus on the technical part of the solution during the discussion and there will be special attention in the formulation of the Definition of Done for each task in the list. Fulfilment of this proposal will be warranted by the Scrum Master which will provide the support and direct the whole agenda in case of any struggles.

3.1.10 Daily Scrum

Participants, which are Product Owner, Scrum Master, and Developers, and everyday occurrence of this short event is correctly justified in the current state and do not require any changes. Content of the meeting which consists of informing about the progress towards the Sprint Goal and state of the Sprint Backlog tasks also fits the definition. The only proposal for improvement of this meeting is the need for Scrum Master’s to focus on the event duration which should not exceed 15 minutes, and which is mostly caused by the extending discussion of the specific topic by a few of the participants. In this kind of situation, when Scrum Master will notice that participants are beginning with a deep discussion that other participants do not need to follow, Scrum Master should interrupt the discussants and advise them to continue with the discussion out of the Daily Scrum event.

3.1.11 Sprint Review

Sprint Review will consist of two parts within the proposed solution. The first part will be dedicated to the presentation of the work that was done during the terminating Sprint. This part will be accomplished with the presence of all four Development teams, their two Scrum Masters, both Product Owners, and all other interested stakeholders such as management of the company, teams that are implementing other modules of the product, customers, etc. The style of the presentation should be in form of a demo presentation, which will contain an explanation of the implemented functionality, its benefits towards the product, an example of a use case, or a live demonstration using the testing hardware. With the consideration of the different types of listeners within the audience of the presentation, the explanation should not contain deep information including technical details of the implementation. Of course, in case of curiosity about this kind of information among the audience, these details can be added to the presentation. During this part of the Sprint Review, there will be also space for a gathering of suggestions originating from the stakeholders and all participants. These comments can be then used within the second part of the Sprint Review.

The agenda of the second part of this event will be a modification of the plan and Product Backlog. Since each team has its own Product Backlog and plans how to achieve the fulfilment of it, this part needs to be done separately by each team. Participants of this part of the Sprint Review will be Developers from the particular team, Scrum Master of the team, and Product Owner. They should all together adjust the Product Backlog by reviewing tasks that were delivered or were not delivered in currently finishing Sprint and using comments that were gathered in the first part of the Sprint Review.

Both proposed parts of the Sprint Review need a person in the role of the moderator, which will be again Scrum Masters. They will control the fulfilment of proposed changes and drive the meeting.

3.1.12 Sprint Retrospective

According to the subsection of the Analysis summary that is discussing Sprint Retrospective 2.6, the current form of this event, which is described in section 2.5.5, follows the definition of the SCRUM. The list of participants, which in the current state consists of Developers and Scrum Master, needs to be extended by the role of Product Owner, who is also participating in the process, and therefore he or she can have a need to share his or her personal opinions and suggestions. Placement of the Sprint Retrospective can be on the last day of a Sprint, as is already proposed in the subsections dedicated to the Sprint 3.1.8. Another thing that can be improved is the way how are results of this meeting processed and implemented into the processes inside the company. If the management wants to gain better results, they need to pay more attention to the needs of their workers which are recorded in this event. This can be achieved by promotion of the gathered list of needs to the company's management by the Scrum Master.

In order to gather more insights and encourage participants to share their ideas, Sprint Retrospective can be done in the form of a game. This proposal of an interactive Retrospective is based on the Playbook [7]. In the phase of preparation, teams that have some members working remotely should create a new collaboration document. In case of all team members can attend this event in person, the team will need space with an available whiteboard. Selected document or whiteboard needs to be divided into three columns with headlights 'What we did well', 'What we can do better', and 'Actions'. This preparation should be followed by a quick revision of the topics from the last Retrospective to build continuity. The facilitator who is Scrum Master should remind these facts to the participants - do not make it personal, and take it personally, listen with an open mind, focus on improvement, the experience of everyone is valid. Then team members can start with writing down what the team did well. This should be in form of one idea per note. Then posted notes need to be grouped by similar and duplicates should be removed. The team should briefly discuss each note. Then the same approach will be applied for the column 'What we can do better', into which will team members post their ideas on what can be improved. Again, the team needs to discuss each topic. Then everyone should brainstorm actions that can be done to improve identified weak areas. Participants again write one action per note, group them and remove duplicates as before. Gathered actions must be discussed within the team, and then they have to be assigned to the members that will take the actions.

3.1.13 Product Backlog

Representation of this artifact using the JIRA system was evaluated as a correct option towards reaching needed Product Backlog characteristics such as transparency, accessibility, visibility, and so on. The area, which was considered as a place that may be improved within the part of the Analysis summary aimed at Product Backlog 2.6, is the creation of new Product Backlog items. As it was explained within the section of analysis which describes current features of the Product Backlog 2.5.6, everyone with the access rights to the JIRA system, has also the possibility to create tasks and therefore Product Backlog items. The possibility of a task creation should be kept in the proposed solution because the form and way how are change request incoming to the Development process 2.3 requires it. What needs to be adjusted is a way how these newly created tasks becoming members of Product Backlog. There can be added either a JIRA label item or a JIRA task state, which would differentiate a newly created task that is not yet part of the Product Backlog from a Product Backlog item. With the addition of this new marker, the Product Owner would have the possibility to evaluate all incoming tasks and would be able to decide about their presence in the Product Backlog. This will reduce the existence of task duplicity, incorrect addressing of the task among the teams, and in case of wrongly or incompletely formulated task, the record will be very easy to ask for additional information.

3.1.14 Sprint Backlog

Taking into account facts written in the part of the analysis describing Sprint Backlog 2.5.7, and subsection of Analysis summary that highlights shortcomings of this SCRUM artifact 2.6, it is more than required to use only one representation of the Sprint Backlog. According to the capabilities of JIRA task records which are currently used as one of two existing Sprint Backlog representations, and which can simply store priority, description, status, and assignee of the task, this will be the one and only representation of the Sprint Backlog.

The presence of a particular task within the Sprint Backlog can be specified by the JIRA label item as it is in the current state. But only Developers of each team must have the right to labelled tasks with this label and therefore include them into the Sprint Backlog of their team. This step of the association of tasks with the Sprint Backlog will be done within the proposed solution after the approval of the task as a Product Backlog item, which will be done by Product Owners and which is explained in the above subsection 3.1.13.

3.2 Financial evaluation of the proposal

Application of the proposal described in the above section 3.1 will be reflected within the firm costs in two ways. The first part of the costs will be related to the existence of additional employees within the proposed solution compared to the actual state of human resources on the project. The second part of the cost will be allocated for the education of the employees in the SCRUM methodology.

Prices of project members labour

For the estimation of the salaries will be used web page *glassdoor.com* [1], which contains information about average salaries for specified positions and specified regions. This source is used also because of the reason that company is not publishing the salaries or any other details of their employees. For the Developers, we will use the average salary for the position of Software developer in the location Czech Republic, which is according to the motioned source [1] approximately 735 500 Kč per year. This means approximately 61 292 Kč per month for one Developer. People that are employed in the role of the Scrum Master earn in the Czech Republic approximately 70 650 Kč per month. The monthly salary of the Product Owner is on average 78 908 Kč according to the previously mentioned web page [1].

Salaries stated in this subsection in the above paragraph are gross salaries, which means that we need to add to these salaries additional costs that employers pay for social and health insurance. For calculation of this cost will be used online calculator of net salaries in the Czech Republic for the year 2021 [16]. The base of the calculation will be gross salary per month which will be calculated from the gross salary per year. Thus, the gross monthly salary of the Developer is approximately 61 292 Kč, which means for employer costs 82 008 Kč per month. Monthly costs that the firm has to pay in order to provide Scrum Master's net salary 70 650 Kč are 94 530 Kč. For the Product Owner's monthly net salary which is an average 78 908 Kč must the company pay 123 790 Kč.

For the simplicity of the estimation, employees in the role of System Architect will have the same salary as Developers, and Team Leaders will have the same salary as Scrum Masters. A list of the average salaries and their costs from the employer's point of view is for better readability written in the table below 3.2.

Table 3.2: **Table of the monthly salaries of the particular project roles and their costs for employer.** (Source: *glassdoor.com* [1], *vypocet.cz* [16])

Role	Monthly gross salary	Monthly employer's costs	Costs per one man-day
Developer	61 292 Kč	82 008 Kč	4 100 Kč
Scrum Master	70 650 Kč	94 530 Kč	4 727 Kč
Product Owner	78 908 Kč	123 790 Kč	6 190 Kč

Prices of SCRUM courses

Prices of the courses used within this cost evaluation are obtained from a price list that is published on the web pages of the company *scrum.cz* [11], which is the biggest provider of SCRUM courses in the Czech Republic. This company is licensed by *scrum.org* [10]. This company was chosen because of her rating as well as because of her location in the Czech Republic. A list of the courses that were evaluated as relevant for this proposal is summarized in the table below 3.3. The table contains information needed for the financial evaluation of this proposal such as the name of the course, duration of the course in days, price of the course in Czech crowns, set of SCRUM roles or project roles that should be course useful for, and official certification that attendee can obtain after successful passing of assignment at the end of the course (as it is written on the web page of the provide [10], there are two tries to pass the assignment included in the price of course for each participant).

The mentioned company [11] offers two types of courses for the Scrum Masters, which are named Professional Scrum Master and Professional Scrum Master II. Both courses have the same price, 27 000 Kč, and both take two days. At the end of the first-named course, the attendee has the possibility to obtain a certification v Professional Scrum Master I. Bypassing the second-mentioned course, the participant can obtain certification called Professional Scrum Master II. In the list of people that should attend the first course are Scrum Team members, Scrum Masters, and people involved in agile product development. This course is also marked as the best introduction to the SCRUM methodology. The second course is designed for Scrum Masters with experience of at least one year, that have a good understanding of a Scrum Framework and want to deepen their knowledge.

Course provider offers also a course for Product Owners called Professional Scrum Product Owner. This course has the same price as a course for Scrum Masters, which is 27 000 Kč. Also, the duration is two days as in the case of previously described courses. After the successful passing of the examination at the end of the course, the participant will gain certification Professional Scrum Product Owner I. This course is mostly for Product Owners and product managers, Scrum Masters who want to obtain a better understanding of the Product Owner role.

There is also a course that is designed for the teams that want to strengthen their SCRUM knowledge. The course has the lowest price out of all courses described within this section, which is 21 000 Kč. The course takes two days and participants can obtain certification of Professional Scrum Master I after successful passing of assignment at the end of the course.

Table 3.3: **List of selected SCRUM courses that are available on web page *scrum.cz*.** (Source: *scrum.cz* [11])

Name of the course	Duration of the course [days]	Price of the course [Kč/person]	Who should attend the course	Certification
Professional Scrum Master I	2	27 000	Scrum Team members, Scrum Masters, and people involved in agile product development	Professional Scrum Master I
Professional Scrum Master II	2	27 000	Scrum Masters with experience of at least one year, that have a good understanding of a Scrum Framework and want to deepen their knowledge	Professional Scrum Master II
Professional Scrum Product Owner	2	27 000	Product Owners and product managers, Scrum Masters who want to obtain a better understanding of the Product Owner role	Professional Scrum Product Owner I
Applying Professional Scrum	2	21 000	Teams that are starting with the SCRUM or teams that want to strengthen their SCRUM knowledge	Professional Scrum Master I

Cost of the project members' labour

Because of the fact that the company is not publishing salaries of the employees, this financial evaluation uses average salaries for a specific job in the Czech Republic which are more detailly explained in separate subsection 3.2 and which are summarized in the table 3.2.

When we compare team structure in the current state of project 2.4 and the structure described in this proposal 3.1, there are two additional team members in the proposed solution. In concrete, there are two new employees in the role of Scrum Masters, if we expect that four employees that are in roles of Scrum Masters at the current state of the project will be reallocated for the positions of Team Leaders.

This if we will take into consideration the average salaries listed in table 3.2, there will be two new employees with a monthly gross salary of 70 650 Kč. Paying this gross salary to two employees costs the company in total 247 580 Kč per one month. Which is 2 970 960 Kč per year if we don't count bonuses or other benefits that could be reflected within the costs. Thus, the conclusion can be that application of the proposed solution will raise the fixed costs of the company approximately by 2 970 960 Kč annually.

Costs of employees' education

The costs related to the education of employees in the theory of SCRUM methodology will have a one-time character and will be spent at the beginning of the process of proposal implementation into the project environment. The main part of these costs will be taken by courses, that are terminated by the final assignment. In case of participant's success in this assignment, he or she will obtain official SCRUM certification. A list of the particular courses that are relevant for this financial evaluation is summarized in table 3.3.

This part of the proposal contains three variants of employees' education. The first variant focuses on the saving of resources 3.2. The second variant provides a compromise between cost-cutting and retraining of all employees 3.2. The last variant aims at the education of all employees without a focus on costs 3.2. All variants are summarized within the table 3.4.

Variant A

Since all employees which are in the roles of Product Owners in the current state of the project already passed the course, there is no need to retrain them again. Also, we can expect that new employees that will take two newly introduced positions of Scrum Masters will be already educated and certified for this role what implies that the firm can save on costs of their training. Thus, only Developers are project members without any official SCRUM courses. Training of the developers can be done internally without their attendance of the courses provided by external firms. Scrum Masters with the assistance of Product Owners can prepare a training session for them. It is necessary to mention that training provided by official trainers, which are specialized in this, and which have experience with training of others, reach the better quality than internal training. After implementation of this variant, all Scrum Masters will have certification 'Professional Scrum Master I', Product Owner will have certification 'Professional Scrum Product Owner I' and all developers will have internal training. Team Leaders which are in the current state of the project in the role of the Scrum Masters have also the certification of 'Professional Scrum Master I'.

So, if we will estimate the duration of this kind of internal training to two working days, which is the same duration as the length of official courses, which will be led by Scrum Masters and Product Owners for Developers, the costs of this educational process will be approximately 306 068 Kč. This amount was calculated by summarization of the costs of salaries from the table 3.2. There are included costs of salaries for two working days of two Product Owners, two Scrum Masters, and thirty-two Developers (it is an average number of Developers if there are four Development teams, and each consists of 6 to 10 employees).

Variant B

This variant is a compromise between the focus of cost-cutting and training of all team members. In this variant, only team members without official certification which was obtained after the official course will be educated. In the current state, only the developers don't have official certification. This implies that Product Owners, Scrum Masters, and Team Leaders of the proposed solution are already certified. So, there are four Development teams that have from 5-9 Developers (we do not count Team Leaders). Therefore, Developers can attend a course called 'Applying Professional Scrum' described in section 3.2 and motioned in the last row of table 3.3. After successful passing of this course, all Developers should be certified with the certification called 'Professional Scrum Master I'. After this, all SCRUM team members will be certified. In concrete all Developers, Team Leaders, and Scrum Masters will have at least certification 'Professional Scrum Master I' and Product Owner will have certification 'Professional Scrum Product Owner I'.

Costs of this variant can be divided into two categories, the first one includes costs of the courses and the second involves costs of the employees' salaries that will be played to them during the presence on SCRUM courses. Costs of the courses will be 588 000 Kč which reflects the cost of the mentioned course which is 21 000 Kč per person, and the number of developers which is on average 7 Developers in four teams. Costs of the salaries, using data from table 3.2 and considering the number of Developers and course duration, will be 229 600 Kč. So, this variant will cost in total 817 600 Kč.

Table 3.4: **Three variants of employees' education, their description, possibly obtained certifications, and their costs.** (Source: Own creation)

Variant	Description	Certifications	Costs
A	Scrum Masters, Team Leaders, and Product Owners already have certification Developers will take an internal course held by Scrum Masters and Product Owners	Scrum Masters and Team Leaders will have Professional Scrum Master I certification Product Owners will have Professional Scrum Product Owner I certification Developers will have internal training without any certification	306 068 Kč
B	Scrum Masters, Team Leaders, and Product Owners already have certification Developers will take an official course named Applying Professional Scrum	Scrum Masters, Team Leaders, and Developer will have Professional Scrum Master I certification Product Owners will have Professional Scrum Product Owner I certification	817 600 Kč
C	Developers will take an official course named Applying Professional Scrum Scrum Masters will take an official course named Professional Scrum Master II Product Owners will take an official course named Professional Scrum Product Owner I	Team Leaders and Developers will have Professional Scrum Master I certification Scrum Masters will have Professional Scrum Master II certification Product Owners will have Professional Scrum Product Owner I certification	1 086 068 Kč

Variant C

This variant is the most expensive one. It proposes the education of all team members by the attendance of official courses without any consideration of their certification ownership. The goal is the presence of the official certification for all employees and the renewal or strengthening of their SCRUM knowledge by participating in the course. At the end of this educational process of this variant, each Developer (including Team Leaders) will be certified with the official certification 'Professional Scrum Master I', each Scrum Master will own 'Professional Scrum Master II' certificate, and both Product Owners will have 'Professional Scrum Product Owner I' certificate.

Costs of this variant will again be the sum of two parts, costs of courses, and costs of salaries. Courses will cost 672 000 Kč for the education of Developers, 54 000 Kč for the training of Scrum Masters, and 54 000 Kč spent on the courses for Scrum Masters. This makes in total 780 000 Kč for the courses. The cost of the salaries for all mentioned employees spend on the two days course will be 306 068 Kč. This the most expensive variant will therefore cost in total 1 086 068 Kč.

3.3 Benefits of the proposed solution

Implementation of the proposed solution describe within this chapter 3, that aims at the application of SCRUM methodology on a software development project should result in the following benefits:

- **Customers' requirements** – because of the fact that Product Owners are not executing any Backlog tasks in the proposed solution, and they are focused on the maximization of the product value and management of the Product Backlog, this will bring the benefit of more flexible reaction to changes in customers' requirements. Also, this will also improve the quality of customer requirements specification.
- **Planning and delivery** – proposed changes are promoting an empiricist approach in which Developers are estimating the time constraint of the tasks based on their experience. The procedure of planning also empowers Developers to decide what will be delivered and more focus is put on the Definition of Done. This will make the process of planning more efficient which will be reflected in more accurate delivery in terms of the deadlines meeting and requirements fulfilment.
- **Product's quality** – there are several intentions of the proposal which should positively affect quality of the product. More effort spent on the customers' requirements will transform into the functionality that will fulfil the purpose of the product on a higher level. Improvement of the planning process means that Developers will have more time to analyse and implement product changes. Also, fact that team members with the broadest technical and product-related knowledge will be deprived of non-technical tasks will empower the technical state of the product. Besides these factors, there are several less significant factors that will all together end up in the product quality increase.
- **Customers' trust** – quicker and more flexible reaction to the customers' requirements on time delivering and better quality of the delivered product will strengthen the relation that customers have towards the product and the company that is selling the product.
- **Workflow improvement** – a superior division of the responsibilities among the team members, as well as a better structure of sprint with better planning, should bring more organization to the process of the product development.

Particular roles will execute only tasks that they have competencies for, and the tasks will be correctly specified and assigned to the teams. Self-organization of the teams, which is also the goal of the described proposal should boost the motivation of the team members. Changes in the Retrospective will help to erase impediments for the working process.

- **Team members' growth** – the proposed solution should transform the team environment into more open in terms of team members' personal realization. This means that they will have more opportunities to bring up their ideas and proposals, they will have more responsibilities as well as they will be able to learn from each other. This will positively affect their soft skills as well as technical knowledge and hard skills in general.
- **Process efficiency** – with the better assignment of the accountabilities, well-defined structure of the Sprint, strict agenda of the events, and correctly designed artifact, the whole process of the product development benefits in more effective usage of all resources such as financial resources, human resources, time capacities, and hardware resources, and so on.

At the top of all benefits summarized above, there is a financial benefit that is the most important for all kinds of business. By the application of the SCRUM methodology, the company will optimize the process of the product development on the observed project, which will encounter all benefits named in this section. The existence of all these benefits will result in better economical profits of the project.

Conclusion

The goal of the thesis is an application of SCRUM methodology on a software development project in order to optimize the working process.

In the theoretical part of the work, there was described software development life cycle and its phases. This explanation was followed by a description of traditional models of software development life cycle. In particular, there were introduced four traditional software development models – Waterfall Model, Iterative Model, Spiral Model, and V-Model. Each section contains short description of the model, graphical illustration of the model and several pros and cons of the model. In the second half of the theoretical chapter, a closer look was taken at agile software development life cycle models. Crystal, Extreme Programming, Lean Software Development, and Kanban were briefly described and then, their advantages and disadvantages were highlighted. This was followed by a more detailed explanation of the SCRUM. Here the focus was put into the definition of the SCRUM principles, all its roles, existing events of the SCRUM, and its artifacts.

The chapter containing analysis of the contemporary situation was opened by the introduction of the company in which is observed project located. Introduction of the company incorporates information about the company location, list of sectors in which its products are used, number of employees and so on. This section also contains insight into the company's organizational structure. Then the project on which should be SCRUM applied was introduced. There was formulated what is the project from this thesis point of view. There was also a description of the teams on the project, product developed within the project, stakeholders of the project, activities incorporated into the project, etc. The next section describes the process of product development and its phases from a detailed perspective. Each phase has its inputs and outputs, and each role of the project has its specific task among the phases. The roles existing within the project were analysed afterward and their responsibilities towards the task within the development process were mapped. Also, events of the process and artifacts used within the process were evaluated and the whole analysis was closed by the summarization of what needs to be adjusted to correctly apply SCRUM methodology to the project.

Within the proposal of the solution, a new team structure was defined at the beginning of the chapter. Then each role of the new team structure was comprehensively described and responsibilities towards the tasks of the process were correctly assigned to the roles. Then adjustment of the Sprint and its evets were provided. Also, artifacts of the development process and their changes were included in the proposal. This was followed by a financial evaluation of the proposed solution, which was based on the summarization of available educational courses ad their prices as well as costs of the project members' labour. The evaluation concluded that the fixed costs of the company will be increased approximately by 2 970 960 Kč annually, which reflects the proposed change of the team's structure. Evaluation of the costs of employees' education was provided in three different variants. The first variant which is the most cost-saving one requires a one-time investment of 306 068 Kč. The second variant would cost 817 600 Kč, and the last and the most expensive form of the employees' education was evaluated to price 1 086 068 Kč. The end of the proposal highlights the benefits of the proposed solution which are expected in several areas and which all should be reflected in better economical profits of the project.

Bibliography

[1] ©2008-2021 Glassdoor Inc.: *glassdoor.com*. [Online; visited 12.4.2021]. Retrieved from: https://www.glassdoor.com/

[2] ©2008-2021 SonarSource S.A: *SonarQube*. [Online; visited 12.3.2021]. Retrieved from: https://www.sonarqube.org/

[3] ©2011-2018 www.javatpoint.com: *Software Engineering Tutorial*. [Online; visited 18.2.2021]. Retrieved from: https://www.javatpoint.com/software-engineering-tutorial

[4] ©2020 SCHWABER Ken; SUTHERLAND Jeff: *The Scrum Guide*. [Online; visited 18.2.2021]. Retrieved from: https://scrumguides.org/

[5] ©2021 Atlassian: *Confluence*. [Online; visited 12.3.2021]. Retrieved from: https://www.atlassian.com/software/confluence

[6] ©2021 Atlassian: *Jira Software*. [Online; visited 12.3.2021]. Retrieved from: https://www.atlassian.com/software/jira

[7] ©2021 Atlassian: *Retrospective*. [Online; visited 12.4.2021]. Retrieved from: https://www.atlassian.com/team-playbook/plays/retrospective

[8] ©2021 Gerrit: *Gerrit Code Review*. [Online; visited 12.3.2021]. Retrieved from: https://www.gerritcodereview.com/

[9] ©2021 IDG Communications, I.: *CIO from IDG*. [Online; visited 12.4.2021]. Retrieved from: https://www.cio.com/

[10] ©2021 Scrum.org: *Scrum.org - The Home Of Scrum*. [Online; visited 18.2.2021]. Retrieved from: https://www.scrum.org/

[11] ©2021 TAYLLORCOX: *scrum.cz*. [Online; visited 12.4.2021]. Retrieved from: https://www.scrum.cz/

[12] ©2021 Tutorials Point: *SDLC Tutorial*. [Online; visited 18.2.2021]. Retrieved from: https://www.tutorialspoint.com/sdlc/index.htm

[13] ©2001 BECK et al.: *Manifesto for Agile Software Development.* [Online; visited 18.2.2021].
Retrieved from: https://agilemanifesto.org/iso/en/manifesto.html

[14] Conservancy, S. F.: *Git –fast-version-control.* [Online; visited 12.3.2021].
Retrieved from: https://git-scm.com/

[15] MYSLÍN, J.: *Scrum: průvodce agilním vývojem softwaru.* Brno: Computer Press. 2016. ISBN 978-8-251-4650-7.

[16] Výpočet.cz: *Výpočet čisté mzdy v roce 2021.* [Online; visited 12.4.2021].
Retrieved from: https://www.vypocet.cz/cista-mzda

[17] ŠOCHOVÁ, Z.; KUNCE, E.: *Agilní metody řízení projektů.* Brno: Computer Press. 2014. ISBN 978-8-251-4194-6.

Appendices

List of Figures

1.1 **The Software Development Life Cycle and its phases.** (Source: Own Creation) . . . 6
1.2 **The Waterfall model of SDLC.** (Source: Own Creation) . . . 8
1.3 **The Iterative model of SDLC.** (Source: Own Creation) . . . 10
1.4 **The Spiral model of SDLC.** (Source: Own Creation) . . . 11
1.5 **The V-model of SDLC.** (Source: Own Creation) . . . 13
1.6 **Crystal methods and their distribution according to the number of people involved in the project.** (Source: Own Creation) . 16
1.7 **Seven principles of Lean Software Development.** (Source: Own Creation) . . . 19
1.8 **Example of Kanban board.** (Source: Own Creation) . . . 20
1.9 **Scrum Team.** (Source: Own Creation) . . . 23
1.10 **Scrum events.** (Source: Own Creation) . . . 25

2.1 **Simplified organizational structure with focus on Brno division.** (Source: Own Creation) . . . 32
2.2 **EPC diagram of the process of product development.** (Source: Own Creation) . . . 35
2.3 **The team structure of the project.** (Source: Own Creation) . . . 44

3.1 **Proposed team structure of the project.** (Source: Own Creation) 58
3.2 **Sprint and its events.** (Source: Own Creation) . . . 65

List of Tables

2.1 **RACI matrix summarizing roles of the development process and their responsibilities according to the tasks within the process.** (Source: Own Creation) 48

3.1 **RACI matrix containing proposal of roles of the development process and their responsibilities according to the tasks within the process.** (Source: Own Creation) 64
3.2 **Table of the monthly salaries of the particular project roles and their costs for employer.** (Source: *glassdoor.com* [1], *vypocet.cz* [16]) 70
3.3 **List of selected SCRUM courses that are available on web page *scrum.cz*.** (Source: *scrum.cz* [11]) 71
3.4 **Three variants of employees' education, their description, possibly obtained certifications, and their costs.** (Source: Own creation) 73